SIAM J. SCI. COMPUT.
Vol. 41, No. 2, pp. A848–A876

LOW-RANK UPDATES AND A DIVIDE-AND-CONQUER METHOD FOR LINEAR MATRIX EQUATIONS*

DANIEL KRESSNER†, STEFANO MASSEI†, AND LEONARDO ROBOL‡

Abstract. Linear matrix equations, such as the Sylvester and Lyapunov equations, play an important role in various applications, including the stability analysis and dimensionality reduction of linear dynamical control systems and the solution of partial differential equations. In this work, we present and analyze a new algorithm, based on tensorized Krylov subspaces, for quickly updating the solution of such a matrix equation when its coefficients undergo low-rank changes. We demonstrate how our algorithm can be utilized to accelerate the Newton method for solving continuous-time algebraic Riccati equations. Our algorithm also forms the basis of a new divide-and-conquer approach for linear matrix equations with coefficients that feature hierarchical low-rank structure, such as hierarchically off-diagonal low-rank structures, hierarchically semiseparable, and banded matrices. Numerical experiments demonstrate the advantages of divide-and-conquer over existing approaches, in terms of computational time and memory consumption.

Key words. Sylvester equation, Lyapunov equation, low-rank update, divide-and-conquer, hierarchical matrices

AMS subject classifications. 15A06, 93C20

DOI. 10.1137/17M1161038

1. Introduction.

This work is concerned with linear matrix equations of the form

$$AX + XB = C \tag{1}$$

for given matrices $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{m\times m}$, and $C \in \mathbb{C}^{n\times m}$. It is well known that this equation admits a unique solution X if and only if the spectra of A and $-B$ are disjoint. For general coefficient matrices, (1) is usually called the Sylvester equation. In the special case $B = A^*$ and $C = C^*$, (1) is called the Lyapunov equation and its solution can be chosen Hermitian. If, moreover, C is negative semidefinite and A is stable (i.e., its spectrum is contained in the open left half plane), then the solution is positive semidefinite.

We specifically target the setting where both m, n are large and A, B, C admit certain data-sparse representations, such as sparsity or (hierarchical) low rank structures. The need for solving such large-scale linear matrix equations arises in various application fields. In dynamical systems and control, Lyapunov equations arise in model reduction [3], linear-quadratic optimal control [13], and stability analysis [21, 24]. In these applications, it is often but not always the case that C has low rank. Partial differential equations (PDEs) are a frequent source of Sylvester equations, where they typically arise from highly structured discretizations of PDEs with separable coefficients; see [29, 41, 53, 56] for recent examples. Other applications arise from the

*Submitted to the journal's Methods and Algorithms for Scientific Computing section December 12, 2017; accepted for publication (in revised form) February 7, 2019; published electronically March 26, 2019.

http://www.siam.org/journals/sisc/41-2/M116103.html

Funding: The third author's work was partially supported by GNCS projects "Metodi numerici avanzati per equazioni e funzioni di matrici con struttura" and "Tecniche innovative per problemi di algebra lineare."

†EPFL, Lausanne, CH-1015, Switzerland (daniel.kressner@epfl.ch, stefano.massei@epfl.ch).

‡Department of Mathematics, University of Pisa, and ISTI-CNR, Pisa, Italy 56124 (leonardo.robol@isti.cnr.it).

linearization of nonlinear problems, such as stochastic dynamic general equilibrium models in macroeconmics [37].

In this work, we study low-rank updates for Lyapunov and Sylvester equations. Given the solution X_0 of the reference equation

$$A_0X_0 + X_0B_0 = C_0, \tag{2}$$

we aim at computing a correction δX such that $X_0+\delta X$ solves the perturbed equation

$$(A_0 + \delta A)(X_0 + \delta X) + (X_0 + \delta X)(B_0 + \delta B) = C_0 + \delta C, \tag{3}$$

where the perturbations δA, δB, δC all have ranks much smaller than $\min\{m, n\}$. This is not only a natural problem to study but it also occurs in some applications. For example, it arises when optimizing dampers in mechanical models [43] or, as we will see below, in the Newton method for solving Riccati equations. However, we expect, and it will be demonstrated in the second part of this paper, that the availability of a fast technique for computing δX will open up a range of other applications.

The literature is scarce on updates of the form (3). Kuzmanović and Truhar [43] view the left-hand side (3) as a low-rank perturbation $\mathbb{L} + \triangle\mathbb{L}$ of the operator $\mathbb{L} : X_0 \to A_0X_0 + X_0B_0$. In turn, this allows us to apply operator variants of the Sherman–Morrison–Woodbury formula discussed, e.g., in [22, 43, 52]. This approach is mathematically equivalent to applying the standard Sherman–Morrison–Woodbury to the $n^2 \times n^2$ linear system corresponding to (3) and it allows us to deal with a much larger class of perturbations, leaving the realm of Sylvester equations. However, it also comes with the major disadvantage of increasing the ranks significantly. For example, if δA has rank r, then the operator $X \to \delta AX$, with the matrix representation $I_n \otimes A$, has rank rn. This makes it impossible to address large values of n with existing techniques for solving Sylvester equations.

The approach proposed in this work proceeds by subtracting (2) from (3), which gives the correction equation

$$(A_0 + \delta A)\delta X + \delta X(B_0 + \delta B) = \delta C - \delta A \cdot X_0 - X_0 \cdot \delta B. \tag{4}$$

This is again a Sylvester equation, but—in contrast to (3)—the right-hand side always has low rank; it is bounded by $\operatorname{rank}(\delta A) + \operatorname{rank}(\delta B) + \operatorname{rank}(\delta C)$. This allows for the use of large-scale Sylvester solvers tailored to low-rank right-hand sides, such as low-rank ADI and (rational) Krylov subspace methods; see [13, 56] for overviews. These techniques return a low-rank approximation to δX and can potentially address very large values of m, n, as long as the data sparsity of $A_0 + \delta A$ and $B_0 + \delta B$ allows for fast matrix-vector multiplication and/or solution of (shifted) linear systems with these matrices. Let us emphasize that our approach is of little use when the rank of C_0 is at the same level as the ranks of the perturbations or even lower. In this case, it is more efficient to solve (3) directly.

In the second part of this work, we devise fast methods for Sylvester equations with coefficients A, B, C that feature hierarchical low-rank structures. In this work, we focus on hierachically off-diagonal low-rank (HODLR) matrices [2], a special case of hierarchical matrices [34], and hierachically semiseparable (HSS) matrices [59], a special case of $\mathcal{H}^2$-matrices [17]. Both formats include banded matrices as an important special case. In fact, there has been recent work by Haber and Verhaegen [33] that aims at approximating the solution X by a banded matrix for Lyapunov equations with banded coefficients. Palitta and Simoncini [48] consider the approximation

of X by the sum of a banded matrix and a low-rank matrix. Both approaches work well for well-conditioned Lyapunov equations but their memory consumption grows considerably as the condition number increases. As we will see below, this difficulty is avoided when approximating X with a hierarchical low-rank matrix instead, even when the coefficients are banded.

Most existing algorithms for solving Lyapunov or Sylvester equations with hierarchical low-rank structure are based on the matrix sign function iteration [23], exploiting the fact that the iterates can be conveniently evaluated and approximated in these formats. The use of the matrix sign function requires the spectra of A and $-B$ to be not only disjoint but separable by a (vertical) line. Sign function based algorithms have been developed for hierarchical matrices [8, 30], sequentially semiseparable matrices [51], HSS matrices [49], and HODLR matrices [47]. Another, less explored direction is to apply numerical quadrature to an integral representation for X and evaluate the resulting matrix inverses or exponentials in a hierarchical low-rank format; see [26] for an example. All these methods exploit the structure indirectly by performing approximate arithmetic operations in the format. This incurs a repeated need for recompression, which often dominates the computational time.

In this work, we develop a new divide-and-conquer method that directly exploits hierarchical low-rank structure and does not require separability of the spectra of A and $-B$. The main idea of the method is to split the coefficients A, B, C into a block diagonal part and an off-diagonal part. The block diagonal part is processed recursively and the off-diagonal part, which is assumed to be of low rank, is incorporated by solving the correction equation (4).

The rest of this paper is organized as follows. In section 2 we recall theoretical tools from the literature that ensure the low-rank approximability of the solution δX of (4). Section 3 describes in detail the low-rank solver employed for approximating δX. We also discuss how to rephrase the Newton's iteration, for solving continuous-time algebraic Riccati equations (CAREs), as the updating of a matrix equation. Numerical tests regarding this application are reported in section 3.5.1. In section 4 we introduce a divide-and-conquer method for solving linear matrix equations whose coefficients can be hierarchically partitioned as block diagonal plus low-rank matrices. We provide an analysis in the case of coefficients represented in the HODLR and HSS formats. The algorithm is tested on examples coming from the discretization of PDEs and linear-quadratic control problems for second-order models. The results are reported in section 5. Finally, in section 6 we draw conclusions and we comment on some open questions.

2. Low-rank approximability. In this section, we recall existing results indicating when the correction δX to the solution of the perturbed Sylvester equation (3) admits a good low-rank approximation. For this purpose, we write (4) more compactly as

$$A\,\delta X + \delta X\,B = D, \qquad \operatorname{rank}(D) \le s := \operatorname{rank}(A) + \operatorname{rank}(B) + \operatorname{rank}(C). \tag{5}$$

In the following, we say that δX admits an ϵ-approximation of rank k if there is a matrix Y of rank at most k such that $\|\delta X - Y\|_2 \le \epsilon$, where $\|\cdot\|_2$ denotes the matrix 2-norm. Clearly, this is the case if and only if the $(k+1)$th largest singular value $\sigma_{k+1}(\delta X)$ is bounded by ϵ (or the size of δX is smaller than $k+1$).

There are numerous results in the literature on the singular value decay of solutions of equations of the form (5); see [4, 5, 30, 31, 50, 54] for examples. Recent work by Beckermann and Townsend [10] yields a general framework for obtaining such re-

sults. Let $\mathcal{R}_{h,h}$ denote the set of rational functions with numerator and denominator of degree at most h. The proof of [10, Thm. 2.1] shows that for every $r \in \mathcal{R}_{h,h}$ there is a matrix Y_h of rank at most kh such that $\delta X - Y_h = r(A)\,\delta X\, r(-B)^{-1}$, provided that the expression on the right-hand side is well defined. In turn,

$$\sigma_{kh+1}(\delta X) \le \|r(A)\|_2 \|r(-B)^{-1}\|_2 \|X\|_2. \tag{6}$$

To proceed from here, we recall that the numerical range of a matrix A is defined as

$$\mathcal{W}(A) := \Big\{ \frac{x^* A x}{x^* x} \;\Big|\; x \in \mathbb{C}^n \backslash \{0\} \Big\}.$$

THEOREM 2.1. *Consider the Sylvester equation* (5) *and let E and F be disjoint compact sets containing the numerical ranges of A and $-B$, respectively. Then*

$$\frac{\sigma_{1+kh}(\delta X)}{\|X\|_2} \leqslant Z_h(E,F) := K_C \min_{r \in \mathcal{R}_{h,h}} \frac{\max_E |r(z)|}{\min_F |r(z)|},$$

where $K_C = 1$ if A, B are normal matrices and $1 \le K_C \le (1+\sqrt{2})^2$ otherwise.

Proof. The result for normal matrices, which is also covered in [10], follows immediately from (6) by diagonalizing A, B. To address the general case, we use Crouzeix's theorem [20], which implies

$$\begin{aligned} \|r(A)\|_2 &\le (1+\sqrt{2}) \max_{z \in E} |r(z)|, \\ \|r(-B)^{-1}\|_2 &\le (1+\sqrt{2}) \max_{z \in E} |1/r(z)| = (1+\sqrt{2}) \big(\min_{z \in F} |r(z)| \big)^{-1}. \end{aligned}$$

Combined with (6), this completes the proof. ∎

The result of Theorem 2.1 links the (relative) singular values of δX with the quantity $Z_h(E,F)$, usually called the hth Zolotarev number [10]. Intuitively, this number becomes small when E and F are well separated. The case when E, F are intervals is particularly well understood and the considerations from [10, sect. 3] lead to the following result.

COROLLARY 2.2. *Let A, B be Hermitian positive definite matrices with spectra contained in an interval $[a,b]$, $0 < a < b < \infty$. Then the solution δX of* (5) *satisfies*

$$\frac{\sigma_{1+kh}(\delta X)}{\|\delta X\|_2} \leqslant 4\rho^{-h}, \qquad \rho := \exp\left(\frac{\pi^2}{\log(4b/a)} \right). \tag{7}$$

Similar results have been derived in [18, 54]. The inequality (7) implies that δX admits an ϵ-approximation of rank kh with ϵ exponentially decaying to zero as h increases. Moreover, the relative separation b/a of the spectra has a very mild logarithmic influence on the exponential decay rate.

Corollary 2.2 easily extends to the case of diagonalizable coefficients with real spectra [14, Cor. 4.3]. Letting $\kappa_{\mathrm{eig}}(A)$ and $\kappa_{\mathrm{eig}}(B)$ denote the condition numbers of eigenvector matrices of A and B, respectively, one has

$$\frac{\sigma_{1+kh}(\delta X)}{\|\delta X\|_2} \leqslant 4\kappa_{\mathrm{eig}}(A)\kappa_{\mathrm{eig}}(B)\rho^{-h}.$$

When E, F are not intervals, it appears to be difficult to derive bounds for $Z_h(E,F)$ that match (7) in terms of strength and elegance; see [10, 45] and the

references therein. Under reasonable assumptions, $Z_h(E,F)$ can be bounded with a function that depends on the so-called logarithmic capacity of the condenser with plates E and F [27]. In particular, Ganelius in [25] showed the inequality

$$Z_h(E,F) \leqslant \gamma \rho^{-h}, \qquad \rho := \exp\left(\frac{1}{\mathrm{Cap}(E,F)}\right), \tag{8}$$

where the constant γ depends only on the geometry of E and F and $\mathrm{Cap}(E,F)$ denotes the logarithmic capacity of the condenser with plates E and F. The decay rate in (8) is tight, i.e., $\lim_{h\to\infty} Z_h(E,F)^{\frac{1}{h}} = \rho^{-1}$; see [27]. However, the estimation of $\mathrm{Cap}(E,F)$ is strongly problem dependent and its asymptotic behavior, when the plates approach each other, has been the subject of quite recent investigations; see [16] and the references therein.

A rather different approach, for getting singular values inequalities, has been suggested by Grasedyck [28]. Letting Γ_A, Γ_B denote disjoint contours encircling the spectra of A and $-B$, respectively, the solution δX of (5) admits the integral representation

$$\delta X = \frac{1}{4\pi^2}\int_{\Gamma_A}\int_{\Gamma_B} \frac{1}{\xi-\eta}(\xi I - A)^{-1} D (\eta I + B)^{-1} d\xi d\eta.$$

Then Γ_B is split up into s parts such that $\frac{1}{\xi-\eta}$ admits a good semiseparable (polynomial) approximation on each part. Each approximation corresponds to a low-rank matrix and by summing up all s parts of Γ_B one obtains a low-rank approximation of δX. Although this technique is shown to establish exponential decay for certain shapes of Γ_A and Γ_B, a small relative separation may require use of a very large s, resulting in unfavorable decay rates.

3. Updating algorithm and application to algebraic Riccati equations. Algorithm 1 summarizes the procedure outlined in the introduction for updating the solution X_0 of $A_0X_0 + X_0B_0 = C_0$ such that $X_0 + \delta X$ approximates the solution of the perturbed Sylvester equation (3). In the following, we discuss details of the implementation of Algorithm 1 and then present a modification for Lyapunov equations as well as an application to Riccati equations.

Algorithm 1 Strategy for solving $(A_0+\delta A)(X_0+\delta X)+(X_0+\delta X)(B_0+\delta B) = C_0+\delta C$.

procedure update_Sylv($A_0, \delta A, B_0, \delta B, C_0, \delta C, X_0$) ▷ $\delta A, \delta B$ and δC have low rank

1: Compute U, V such that $\delta C - \delta A X_0 - X_0 \delta B = UV^*$
2: $\delta X \leftarrow$ LOW_RANK_SYLV$(A_0 + \delta A, B_0 + \delta B, U, V)$
3: **return** $X_0 + \delta X$

end procedure

3.1. Step 1: Construction of low-rank right-hand side. Given (low-rank) factorizations of the perturbations to the coefficients,

$$\delta A = U_A V_A^*, \quad \delta B = U_B V_B^*, \quad \delta C = U_C V_C^*,$$

a factorization $\delta C - \delta A X_0 - X_0 \delta B = UV^*$ can be cheaply obtained by simply setting

$$U = [U_C, -U_A, -X_0 U_B], \qquad V = [V_C, X_0^* V_A, V_B], \tag{9}$$

where U, V both have $s = \mathrm{rank}(\delta A) + \mathrm{rank}(\delta B) + \mathrm{rank}(\delta C)$ columns.

The computational cost of low-rank solvers for Sylvester equations, such as the extended Krylov subspace method discussed in the next section, critically depends on the rank of the right-hand side. It is therefore advisable to perform a compression of the factors (9) in order to decrease the rank. For this purpose, we compute reduced QR decompositions $U = Q_U R_U$ and $V = Q_V R_V$ such that $Q_U \in \mathbb{R}^{m\times s}$, $Q_V \in \mathbb{R}^{n\times s}$ have orthonormal columns and $R_U, R_V \in \mathbb{R}^{s\times s}$ are upper triangular. We then compute the singular values $\sigma_1, \dots, \sigma_s$ of the $s \times s$ matrix $R_U R_V^*$. We only retain the first $\tilde{s} \le s$ singular values, such that $\sigma_{\tilde{s}+1}/\sigma_1 \le \tau_\sigma$ for a user-specified tolerance $\tau_\sigma > 0$. Letting U_R and V_R contain the first $\tilde{s}$ left and right singular vectors, respectively, and $\Sigma := \operatorname{diag}(\sigma_1, \dots, \sigma_{\tilde{s}})$, we set

$$\widetilde{U} := Q_U U_R \sqrt{\Sigma}, \qquad \widetilde{V} := Q_V V_R \sqrt{\Sigma}.$$

By construction, $\frac{\|\widetilde{U}\widetilde{V}^* - UV^*\|_2}{\|UV^*\|_2} \leqslant \tau_\sigma$ and we can therefore safely replace the factorization UV^* by $\widetilde{U}\widetilde{V}^*$. Dominated by the computation of the two QR decompositions and the SVD, the described compression procedure requires $\mathcal{O}((m+n)s^2 + s^3)$. In our setting, this method is attractive because $s \ll \min\{n, m\}$.

3.2. Step 2: Solution of correction equation. Step 2 of Algorithm 1 requires solving a Sylvester equation of the form

$$AX + XB = UV^*, \tag{10}$$

where $A = A_0 + \delta A$, $B = B_0 + \delta B$, and U, V have $s \ll \min\{n, m\}$ columns. We assume that X can be well approximated by a low-rank matrix, which is the case—for example—when the hypotheses of Theorem 2.1 are satisfied with two sets E and F ensuring a fast decay of $Z_\ell(E, F)$. The most common solvers for (10) are ADI-type methods and Krylov subspace projection algorithms [56]. One particularly effective approach to obtain a low-rank approximation of X is the rational Krylov subspace method [12] with the poles determined, e.g., by the rational approximation from Theorem 2.1. On the other hand, the *extended Krylov subspace method* introduced in [55] does not require the estimation of such parameters and has been observed to be quite effective for many situations of interest. In the following, we therefore use this method for its simplicity but stress that any of the low-rank solvers mentioned above can be used instead.

The extended Krylov subspace method constructs orthonormal bases U_t and V_t of the subspaces

$$\begin{aligned}\mathcal{U}_t &:= \mathcal{EK}_t(A, U) = \operatorname{span}\{U, A^{-1}U, AU, A^{-2}U, \dots, A^{t-1}U, A^{-t}U\},\\ \mathcal{V}_t &:= \mathcal{EK}_t(B^*, V) = \operatorname{span}\{V, (B^*)^{-1}V, B^*V, (B^*)^{-2}V, \dots, (B^*)^{t-1}V, (B^*)^{-t}V\}\end{aligned}$$

for some t satisfying $2ts < \min\{m, n\}$. This is done by means of two extended block Arnoldi processes. Then, the compressed equation

$$\widetilde{A}X_t + X_t\widetilde{B} = \tilde{C}, \quad \widetilde{A} = U_t^* A U_t, \quad \widetilde{B} = V_t^* B V_t, \quad \widetilde{C} = U_t^* U V^* V_t,$$

is solved by the Bartels–Stewart method [7]. Note that the matrices $\widetilde{A}, \widetilde{B}$, and $\widetilde{C}$ do not need to be computed explicitly but can be obtained from the coefficients generated during the extended block Arnoldi processes; see [55, Proposition 3.2] for more details. The matrix $\widetilde{X} = U_t X_t V_t^*$ is returned as approximation to the solution of (10). The resulting method is summarized in Algorithm 2.

We offer a few remarks concerning the implementation of Algorithm 2:

Algorithm 2 Extended Krylov subspace method for solving $AX + XB = UV^*$.

1: **procedure** LOW_RANK_SYLV(A, B, U, V) ▷ U, V have both s columns
2: $[U_1, -] = \mathtt{qr}([U, A^{-1}U])$, $[V_1, -] = \mathtt{qr}([V, A^{-1}V])$
3: **for** $t = 1, 2, \dots$ **do**
4: $\widetilde{A} \leftarrow U_t^* A U_t$, $\widetilde{B} \leftarrow V_t^* B V_t$, $\widetilde{C} \leftarrow U_t^* U V^* V_t$
5: $X_t \leftarrow$ DENSE_SYLV($\widetilde{A}, \widetilde{B}, \widetilde{C}$)
6: **if** Converged **then**
7: **return** $\widetilde{X} = U_t X_t V_t^*$
8: **end if**
9: Select the last $2s$ columns: $U_t = [U^{(0)}, U^{(+)}, U^{(-)}]$, $V_t = [V^{(0)}, V^{(+)}, V^{(-)}]$
10: $\widetilde{U} = [AU^{(+)}, A^{-1}U^{(-)}]$, $\widetilde{V} = [B^*V^{(+)}, (B^*)^{-1}V^{(-)}]$
11: $\widetilde{U} \leftarrow \widetilde{U} - U_t U_t^* \widetilde{U}$, $\widetilde{V} \leftarrow \widetilde{V} - V_t V_t^* \widetilde{V}$ ▷ Orthogonalize w.r.t. U_t and V_t
12: $[\tilde{U}, -] = \mathtt{qr}(\tilde{U})$, $[\tilde{V}, -] = \mathtt{qr}(\tilde{V})$
13: $U_{t+1} = [U_t, \tilde{U}]$, $V_{t+1} = [V_t, \tilde{V}]$
14: **end for**
15: **end procedure**

- The number of iterations t is chosen adaptively to ensure that the relation

$$\|A\widetilde{X} + \widetilde{X}B - UV^*\|_2 \leqslant \tau \tag{11}$$

  is satisfied for some tolerance τ, which is chosen to be small relative to the norm of $\widetilde{X}$, i.e., $\tau = \tau_{\mathsf{EK}} \cdot \|\widetilde{X}\|_2$ for a small τ_{EK}. This relation can be efficiently verified as described in [35, 55].
- The matrices generated in line 11 of Algorithm 2 might become numerically rank deficient. Several techniques have been proposed to deal with this phenomenon; see [15, 32] and the references therein. We use the strategy described in [15, Algorithm 7.3], which performs pivoted QR decompositions in line 13 and only retains columns corresponding to nonnegligible diagonal entries in the triangular factors. This reduces the size of the block vectors in all subsequent steps.
- For applying A^{-1}, B^{-1} in Algorithm 2, (sparse) LU factorizations of A and B are computed once before starting the extended block Arnoldi process.
- When the algorithm is completed, we perform another compression of the returned solution by computing the truncated SVD of X_t and using the same threshold τ_σ employed for compressing the right-hand side.

3.3. Residual. As the correction equation is solved iteratively and inexactly, until the stopping criterion (11) is satisfied, it is important to relate the accuracy of the solution X to the accuracy of X_0 and δX. Suppose that the computed approximations $\hat{X}_0$ and $\delta\hat{X}$ satisfy

$$\|A_0\hat{X}_0 + \hat{X}_0 B_0 - C_0\| \leqslant \tau_0, \quad \|(A_0+\delta A)\delta\hat{X}_0 + \delta\hat{X}_0(B_0+\delta B) - (\delta C - \delta A\hat{X}_0 - \hat{X}_0\delta B)\| \leqslant \tau_\delta.$$

By the triangular inequality, we can then conclude that $\hat{X} = \hat{X}_0 + \delta\hat{X}$ satisfies

$$\|(A_0 + \delta A)\hat{X} + \hat{X}(B_0 + \delta B) - (C_0 + \delta C)\| \leqslant \tau_0 + \tau_\delta.$$

Hence, in order to avoid unnecessary work, it is advisable to choose τ_δ not smaller than τ_0.

3.4. Stable Lyapunov equations. We now consider the special case of a Lyapunov equation $A_0X_0 + X_0A_0^* = C_0$, where $A_0 \in \mathbb{C}^{n\times n}$ is stable and $C_0 \in \mathbb{C}^{n\times n}$ is Hermitian negative semidefinite. We assume that the perturbed equation $A(X_0 + \delta X) + (X_0 + \delta X)A^* = C_0 + \delta C$, with $A = A_0 + \delta A$, has the same properties, implying that both X_0 and $X_0 + \delta X$ are Hermitian positive semidefinite. In general, this does *not* ensure that the corresponding correction equation

$$A\,\delta X + \delta X\,A^* = \delta C - \delta A\,X_0 - X_0\,\delta A^* \tag{12}$$

inherits these properties. In particular, the right-hand side may be indefinite. In turn, large-scale solvers tailored to stable Lyapunov equations with low-rank right-hand side [46] cannot be directly applied to (12). Following [11, sect. 2.3.1], this issue can be addressed by splitting the right-hand side.

To explain the idea of splitting, suppose we have low-rank factorizations $\delta A = U_A V_A^*$, $\delta C = U_C \Sigma_C U_C^*$, with Σ_C Hermitian. In turn, the right-hand side of (12) can be written as

$$\delta C - \delta A X_0 - X_0 \delta A^* = \widetilde{U}\Sigma\widetilde{U}^* \quad \text{with} \quad \widetilde{U} = [U_C, U_A, X_0V_A], \quad \Sigma = \begin{bmatrix} \Sigma_C & & \\ & & -I \\ & -I & \end{bmatrix}.$$

After computing a reduced QR factorization $\widetilde{U} = Q_{\widetilde{U}}R_{\widetilde{U}}$, we compute a (reduced) spectral decomposition of $R_{\widetilde{U}}\Sigma R_{\widetilde{U}}^*$ such that

$$R_{\widetilde{U}}\Sigma R_{\widetilde{U}}^* = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} D_1 & 0 \\ 0 & -D_2 \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^*,$$

where D_1, D_2 are diagonal matrices with positive diagonal elements. After setting $U_1 = Q_{\widetilde{U}}Q_1\sqrt{D_1}$, $U_2 = Q_{\widetilde{U}}Q_2\sqrt{D_2}$, this allows us to write $R_{\widetilde{U}}\Sigma R_U^* = U_1U_1^* - U_2U_2^*$. Hence, after solving the two stable Lyapunov equations

$$A\,\delta X_1 + \delta X_1\,A^* = -U_1U_1^*, \qquad A\,\delta X_2 + \delta X_2\,A^* = -U_2U_2^*, \tag{13}$$

the solution of (12) is obtained as $\delta X = \delta X_2 - \delta X_1$.

The extended Krylov subspace method applied to (13) operates with the subspaces $\mathcal{EK}_t(A, U_1)$ and $\mathcal{EK}_t(A, U_2)$. This is more favorable than a naive application of the method to the original nonsymmetric factorization

$$[U_C, -U_A, -X_0V_A][U_C, X_0V_A, U_A]^*,$$

which would operate with two subspaces of double dimension.

Another approach, which does not require the splitting of the right-hand side, relies on projecting the Lyapunov equation onto the extended Krylov subspace $\mathcal{EK}(A, \widetilde{U})$. In this way, we only need to generate a single Krylov subspace, even though with a larger block vector. In our experience, the performances of the two approaches are analogous.

3.5. Solving algebraic Riccati equation by the Newton method. We now present a practical situation that requires solving several perturbed Lyapunov equations sequentially.

Consider the CARE

$$AX + XA^* - XBX = C, \tag{14}$$

where $A \in \mathbb{C}^{n\times n}$ is a general matrix, while $B \in \mathbb{C}^{n\times n}$ is Hermitian positive semidefinite and $C \in \mathbb{C}^{n\times n}$ is Hermitian negative semidefinite. We also assume that the pair (A^*, B) is *stabilizable*, i.e., there exists $X_0 \in \mathbb{C}^{n\times n}$ such that $A - X_0B$ is stable. Moreover, we suppose that (C, A^*) is detectable, that is, equivalent to the stabilizability of (A, C^*). Under these assumptions, there exists a unique Hermitian positive semidefinite solution $X \in \mathbb{C}^{n\times n}$ of (14) such that $A - XB$ is stable [42]. This is called the *stabilizing* solution.

Kleinman's formulation [38] of the Newton method (14) requires solving the matrix equation

$$(A - X_kB)X_{k+1} + X_{k+1}(A^* - BX_k) = C - X_kBX_k \tag{15}$$

for determining the next iterate X_{k+1}. Assuming that the starting matrix X_0 is Hermitian, the matrices X_k are Hermitian too and (15) is a Lyapunov equation with Hermitian right-hand side.

Under mild hypotheses, any Hermitian starting matrix X_0 such that $A - X_0B$ is stable yields a quadratically convergent Hermitian sequence whose limit is the stabilizing solution [44]. Moreover, the sequence is nonincreasing in terms of the Loewner ordering. If A is already stable, a natural choice is $X_0 = 0$.

In many examples of linear-quadratic control problems [1], the coefficient B has low-rank, i.e., it takes the form $B = B_UB_U^*$, where B_U only has a few columns. In turn, two consecutive equations (15) can be linked via low-rank updates. More explicitly, (15) can be rewritten as

$$\begin{aligned}(A_{k-1} + (X_{k-1} - X_k)B)X_{k+1} + X_{k+1}(A_{k-1}^* + B(X_{k-1} - X_k))\\ = C_{k-1} + X_{k-1}BX_{k-1} - X_kBX_k,\end{aligned}$$

where $A_{k-1} := A - X_{k-1}B$ and $C_{k-1} := C - X_{k-1}BX_{k-1}$.

Thus, after the—possibly expensive—computation of X_1 one can compute the updates $\delta X_k := X_{k+1} - X_k$ by solving

$$(A - X_kB)\delta X_k + \delta X_k(A^* - BX_k) = \delta X_{k-1}B\delta X_{k-1}, \quad k = 1, 2, \dots. \tag{16}$$

For this purpose, we use a variant of Algorithm 2 tailored to Lyapunov equations, denoted by LOW_RANK_LYAP. In contrast to the more general situation discussed in section 3.4, the right-hand side is always positive semidefinite and therefore no splitting is required. The resulting method is described in Algorithm 3. For large-scale problems, matrix A_k at line 7 is not formed explicitly and we rely on the Sherman–Morrison–Woodbury formula to compute the action of A_k^{-1}. If the final solution is rank structured, then recompression is advised after the sum at line 10.

3.5.1. Numerical experiments. We demonstrate the performance of Algorithm 3 with two examples. Details of the implementation, the choice of parameters, and the computational environment are discussed in section 5.1 below.

Example 3.1. We first consider the convective thermal flow problem from the benchmark collection [40], leading to a CARE with coefficients $A \in \mathbb{R}^{9669\times 9669}$, $B_U \in \mathbb{R}^{9669\times 1}$, $C = -C_UC_U^*$ for $C_U \in \mathbb{R}^{9669\times 5}$. The matrix A is symmetric negative definite and sparse, and only 67391 entries are nonzero. When applying the standard Newton method to this CARE, the right-hand side of the Lyapunov equation (15), which needs to be solved in every step, has rank at most 6. On the other hand, the right-hand side of (16) has rank 1. As the computational effort of low-rank solvers for

Algorithm 3 Low-rank update Newton method for solving $AX+XA^*-XB_UB_U^*X = C$.

procedure NEWT_RICCATI(A, B_U, C, X_0)
  $A_0 \leftarrow A - X_0 B_U B_U^*$
  $C_0 \leftarrow C - X_0 B_U B_U^* X_0$
  $X_1 \leftarrow$SOLVE_LYAP(A_0, C_0) ▷ Any Lyapunov solver
  $\delta X_0 \leftarrow X_1 - X_0$
  for $k = 1, 2, \ldots$ **do**
    $A_k \leftarrow A - X_k B_U B_U^*$
    $C_U \leftarrow \delta X_{k-1} B_U$
    $\delta X_k \leftarrow$ LOW_RANK_LYAP(A_k, C_U)
    $X_{k+1} = X_k + \delta X_k$
    if $\|\delta X_k\|_2 < \tau_{\mathsf{NW}} \cdot \|X_1\|_2$ **then**
      return X_{k+1}
    end if
  end for
end procedure

TABLE 1
Performance of Algorithm 3 *and the standard Newton method for the CARE from Example* 3.1.

	Algorithm 3						Standard Newton method			
n	$\|\hat{X}\|_2$	T_{tot}	$\frac{T_{\text{step 1}}}{T_{\text{tot}}}$	T_{avg}	Res	it	T_{tot}	T_{avg}	Res	it
9,669	$2.96 \cdot 10^1$	53.35	0.14	4.16	$1.21 \cdot 10^{-7}$	12	89.84	7.49	$1.65 \cdot 10^{-7}$	13

Lyapunov equations typically grows linearly with the rank, this makes Algorithm 3 more attractive. In particular, this is the case for the extended Krylov subspace method, Algorithm 2, used in this work.

The performance of both variants of the Newton method is reported in Table 1. While T_{tot} denotes the total execution time (in seconds), T_{avg} denotes the average time needed for solving the Lyapunov equation in every step of the standard Newton method or Algorithm 3 (after the first step). The quantity $\frac{T_{\text{step 1}}}{T_{\text{tot}}}$ shows the fraction of time spent by Algorithm 3 on the (more expensive) first step. *it* refers to the number of iterations and *Res* refers to the relative residual $\|A\hat{X} + \hat{X}A^* - \hat{X}B\hat{X} - C\|_2/\|AX_0 + X_0A^* - X_0BX_0 - C\|_2$ of the approximate solution $\hat{X}$ returned by each of the two variants. The results reveal that Algorithm 3 is faster while delivering the same level of accuracy.

Example 3.2. We now consider Example 4.3 from [1], a CARE with coefficients

$$A = \begin{bmatrix} 0 & -\frac{1}{4}K \\ I_q & -I_q \end{bmatrix}, \quad K = \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{q\times q},$$

$$B = B_U B_U^*, \qquad B_U = \begin{bmatrix} 0 \\ \frac{1}{4}D \end{bmatrix} \in \mathbb{R}^{2q\times 2}, \qquad D = \begin{bmatrix} e_1 & e_q \end{bmatrix} \in \mathbb{R}^{q\times 2}, \quad C = I_{2q},$$

where e_j denotes the jth unit vector of length q. Except for one zero eigenvalue, the spectrum of A is contained in the open left half plane. A stabilizing initial guess is

TABLE 2
Performance of Algorithm 3 and the standard Newton method for the CARE from Example 3.2.

	Algorithm 3						Standard Newton method			
n	$\|\hat{X}\|_2$	T_{tot}	$\frac{T_{\text{step 1}}}{T_{\text{tot}}}$	T_{avg}	Res	it	T_{tot}	T_{avg}	Res	it
512	$1.55 \cdot 10^4$	1.21	0.39	0.07	$3.42 \cdot 10^{-9}$	11	5.37	0.49	$1.69 \cdot 10^{-13}$	12
1,024	$6.19 \cdot 10^4$	6.23	0.84	0.09	$1.52 \cdot 10^{-8}$	12	53.18	4.43	$4.70 \cdot 10^{-13}$	13
1,536	$1.39 \cdot 10^5$	22.7	0.91	0.18	$3.05 \cdot 10^{-8}$	12	252.58	21.05	$7.95 \cdot 10^{-13}$	13
2,048	$2.48 \cdot 10^5$	62.37	0.96	0.25	$5.78 \cdot 10^{-8}$	12	735.99	61.33	$1.34 \cdot 10^{-12}$	13

given by

$$X_0 = EE^*, \quad E = 2\begin{bmatrix} -e_q & e_1 \\ -e_q & e_1 \end{bmatrix} \in \mathbb{R}^{2q \times 2}.$$

Note that C has full rank, which prevents us from the use of low-rank solvers for addressing the Lyapunov equation (15) in the standard Newton iteration and we need to resort to the (dense) Bartels–Stewart method implemented in the MATLAB function `lyap`. In contrast, the right-hand side of (16) has rank 2, which allows us to use such low-rank solvers in every but the first step of Algorithm 3.

The obtained results, for varying $n := 2q$, are shown in Table 2. Not surprisingly, Algorithm 3 is much faster in this example because it only relies on the dense solver in the first step.

4. A divide-and-conquer approach. In this section we use low-rank updates to devise a new divide-and-conquer method for the Sylvester equation (1). For simplicity, we consider the case $m = n$ and hence the solution X is a square $n \times n$ matrix. In principle, our developments extend to the case $m \neq n$ but additional technicalities come into play, e.g., the hierarchical low-rank formats need to be adjusted.

Suppose that the coefficients of (1) can be decomposed as

$$A = A_0 + \delta A, \qquad B = B_0 + \delta B, \qquad C = C_0 + \delta C, \tag{17}$$

where A_0, B_0, C_0 are block diagonal matrices of the same shape and the corrections $\delta A, \delta B, \delta C$ all have low rank. This is, in particular, the case when all coefficients are banded but (17) allows us to handle more general situations. We apply Algorithm 1 to deal with the low-rank corrections. It thus remains to solve the smaller Sylvester equations associated with the diagonal blocks of $A_0X_0 + X_0B_0 = C_0$. If the diagonal blocks of A_0, B_0, C_0 again admit a decomposition of the form (17), we can recursively repeat the procedure. Keeping track of the low-rank corrections at the different levels of the recursions requires us to work with hierarchical low-rank formats, such as the HODLR and the HSS formats.

4.1. HODLR matrices. A HODLR matrix $A \in \mathbb{C}^{n \times n}$, as defined in [2, 6], admits a block partition of the form

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \tag{18}$$

where A_{12}, A_{21} have low rank and A_{11}, A_{22} are again of the form (18). This recursive partition is continued until the diagonal blocks have reached a certain minimal block size. For later purposes, it is helpful to give a formal definition of the HODLR format

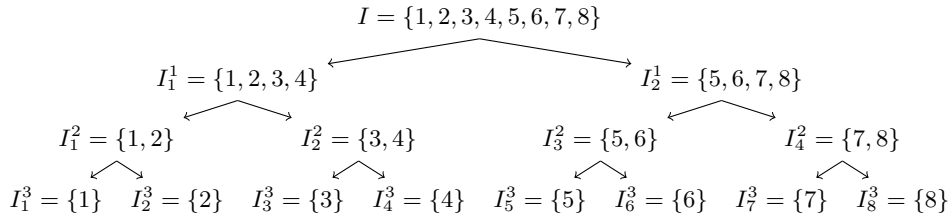

FIG. 1. *Example of a cluster tree of depth* 3.

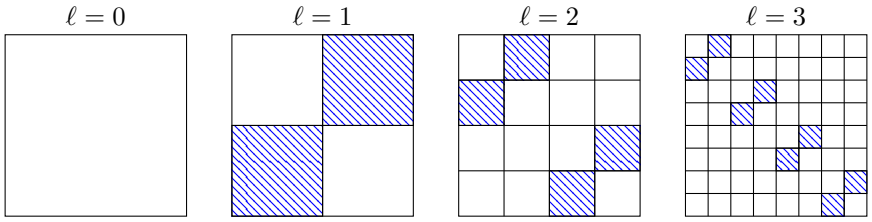

FIG. 2. *Block partitions induced by each level of a cluster tree of depth* 3.

that proceeds in the opposite direction, from the finest block level to the full matrix. Given an integer p, let us consider an integer partition

$$n = n_1 + n_2 + \cdots + n_{2^p}, \tag{19}$$

where p and n_i are usually chosen such that all n_i are nearly equal to the minimal block size. We build a perfectly balanced binary tree, the so-called cluster tree, from this partition by setting $n_i^{(p)} := \sum_{j=1}^{i} n_j$ and defining the leaf nodes to be

$$I_1^p = \{1, \ldots, n_1^{(p)}\}, \quad I_2^p = \{n_1^{(p)} + 1, \ldots, n_2^{(p)}\}, \quad \ldots, \quad I_{2^p}^p = \{n_{2^p-1}^{(p)} + 1, \ldots, n\}.$$

The nodes of depth $\ell < p$ are defined recursively by setting $I_i^\ell = I_{2i-1}^{\ell+1} \cup I_{2i}^{\ell+1}$ for $i = 1, \ldots, 2^\ell$. At the root, $I_1^0 = I := \{1, \ldots, n\}$.

Figure 1 provides an illustration of the cluster tree obtained for $n = 8$, with the (impractical) minimal block size 1. We use $\mathcal{T}_p$ to denote the cluster tree associated with (19).

There are 2^ℓ nodes on level ℓ of $\mathcal{T}_p$ and they partition a matrix $A \in \mathbb{C}^{n \times n}$ into a $2^\ell \times 2^\ell$ block matrix with the blocks $A(I_i^\ell, I_j^\ell)$ for $i, j = 1, \ldots, 2^\ell$. For the HODLR format, only the off-diagonal blocks appearing in the recursive partition (18) are relevant. These are given by

$$A(I_i^\ell, I_j^\ell) \quad \text{and} \quad A(I_j^\ell, I_i^\ell) \quad \text{for} \quad (i,j) = (2,1), (4,3), \ldots, (2^\ell, 2^\ell - 1), \quad \ell = 1, \ldots, p, \tag{20}$$

and marked with stripes in Figure 2.

DEFINITION 4.1. *Consider a cluster tree $\mathcal{T}_p$ and let $A \in \mathbb{C}^{n \times n}$. Then*

1. *for $k \in \mathbb{N}$, A is said to be a $(\mathcal{T}_p, k)$-HODLR matrix if each of the off-diagonal blocks listed in* (20) *has rank at most k;*
2. *the HODLR rank of A (with respect to $\mathcal{T}_p$) is the smallest integer k such that A is a $(\mathcal{T}_p, k)$-HODLR matrix.*

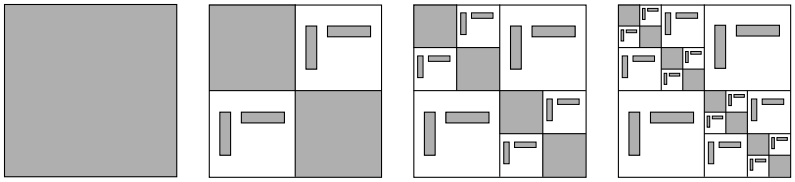

FIG. 3. *Pictorial description of the HODLR format for cluster trees of different depths. The subblocks filled in gray are dense blocks; the others are stored as low-rank outer products.*

A $(\mathcal{T}_p, k)$-HODLR matrix A can be stored efficiently by replacing each off-diagonal block $A(I_i^\ell, I_j^\ell)$ featuring in Definition 4.1 by its low-rank factorization $U_i^{(\ell)}\big(V_j^{(\ell)}\big)^*$. Both, $U_i^{(\ell)}$ and $V_j^{(\ell)}$ have at most k columns. In turn, the only full blocks that need to be stored are the diagonal blocks at the lowest level: $A(I_i^p, I_i^p)$ for $i = 1, \dots, 2^p$; see also Figure 3. If $n = 2^p k$ and $n_i = k$ in (19), $O(nk)$ memory is needed for storing the diagonal blocks as well as the low-rank factors on each level ℓ. Hence, the storage of such a HODLR matrix requires $O(pnk) = O(kn \log n)$ memory in total.

4.2. Divide-and-conquer in the HODLR format. We are now prepared to describe the divide-and-conquer method for a Sylvester equation (1) with $(\mathcal{T}_p, k)$-HODLR matrices A, B, C. By definition, we can write

(21)
$$A = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} + \begin{bmatrix} 0 & A_{12} \\ A_{21} & 0 \end{bmatrix} = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} + \begin{bmatrix} 0 & U_1 V_2^* \\ U_2 V_1^* & 0 \end{bmatrix} = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} + U_A V_A^*,$$

where

(22)
$$U_A := \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix}, \qquad V_A := \begin{bmatrix} 0 & V_1 \\ V_2 & 0 \end{bmatrix}$$

have at most $2k$ columns each. The matrices U_B, V_B, U_C, V_C are defined analogously. Note that the diagonal blocks are $(\mathcal{T}_{p-1}, k)$-HODLR matrices for a suitably defined cluster tree $\mathcal{T}_{p-1}$. After solving (recursively) for these diagonal blocks, we apply the technique described in Algorithm 1 to incorporate the low-rank corrections for A, B, C. The described procedure is summarized in Algorithm 4.

When implementing Algorithm 4 it is advisable to recompress the right-hand side UV^*, obtained in line 9, using the procedure described in section 3.1. Similarly, line 11 aims at recompressing the entire HODLR matrix to mitigate the increase of the HODLR rank due to the addition of δX. For this purpose, the procedure from section 3.1 is applied, with truncation tolerance τ, to each off-diagonal block (20). This step is optional because it is expensive and we cannot expect a significant rank reduction for Sylvester equations with general HODLR coefficients; see also Remark 4.4 below.

When Algorithm 4 is applied to a stable Lyapunov equation, the techniques from section 3.4 are employed in line 10 in order to preserve the symmetry of δX. Note, however, that Algorithm 4 does not preserve definiteness, that is, δX is, in general, not positive semidefinite. We pose it as an open problem to design a divide-and-conquer method that has this desirable property, implying that the solution is approached monotonically from below.

4.2.1. A priori bounds on the HODLR rank of X. The numerical rank of the correction δX, computed in line 10, can be bounded using the tools introduced in section 2.

Algorithm 4 Divide-and-conquer method for Sylvester equations with HODLR coefficients.

procedure D&C_SYLV(A, B, C) ▷ Solve $AX + XB = C$
  if A, B are dense matrices **then**
    return DENSE_SYLV(A, B, C)
  else
    Decompose

$$A = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} + U_A V_A^*,\ B = \begin{bmatrix} B_{11} & 0 \\ 0 & B_{22} \end{bmatrix} + U_B V_B^*,\ C = \begin{bmatrix} C_{11} & 0 \\ 0 & C_{22} \end{bmatrix} + U_C V_C^*$$

    with $U_A, V_A, U_B, V_B, U_C, V_C$ defined as in (22).
    $X_{11} \leftarrow$ D&C_SYLV(A_{11}, B_{11}, C_{11})
    $X_{22} \leftarrow$ D&C_SYLV(A_{22}, B_{22}, C_{22})
    Set $X_0 \leftarrow \begin{bmatrix} X_{11} & 0 \\ 0 & X_{22} \end{bmatrix}$
    Set $U = [U_C, -U_A, -X_0 U_B]$ and $V = [V_C, X_0^* V_A, V_B]$.
    $\delta X \leftarrow$ LOW_RANK_SYLV(A, B, U, V)
    return COMPRESS($X_0 + \delta X, \tau$) ▷ Compression is optional
  end if
end procedure

LEMMA 4.2. *Let $A, B, C \in \mathbb{C}^{n\times n}$ be $(\mathcal{T}_p, k)$-HODLR matrices and suppose that $\mathcal{W}(A) \subseteq E$, $\mathcal{W}(-B) \subseteq F$ for sets $E, F \subset \mathbb{C}$ satisfying $E \cap F = \emptyset$, and run Algorithm 4 with input arguments A, B, and C. Then for every recursion of Algorithm 4, the correction δX satisfies*

$$\frac{\sigma_{6kh+1}(\delta X)}{\|\delta X\|_2} \leqslant \left(1 + \sqrt{2}\right) \cdot Z_h(E, F). \tag{23}$$

Proof. As the matrices A and B appearing in each recursion of Algorithm 4 are principal submatrices of the input matrices A and B, their numerical ranges are contained in E and $-F$, respectively. Moreover, the rank of the right-hand-side UV^* is bounded by $6k$. Thus, applying Theorem 2.1 to $A\,\delta X + \delta X\, B = UV^*$ establishes the claim. ∎

We now use Lemma 4.2 to derive an a priori approximation result for X. Let $\delta X_\ell \in \mathbb{C}^{n\times n}$ be the block diagonal matrix for which the diagonal blocks contain all corrections computed at recursion level $\ell \leq p-1$ of Algorithm 4. Note that the block partitioning of δX_ℓ corresponds to level ℓ of $\mathcal{T}_p$; see also Figure 2. Similarly, let $X_0 \in \mathbb{C}^{n\times n}$ be the block diagonal matrix that contains all the solutions of dense Sylvester equations at level p. Then $X = X_0 + \delta X_0 + \cdots + \delta X_{p-1}$. Given $\tilde{\epsilon} > 0$, suppose that h is chosen such that $(1+\sqrt{2})Z_h(E, F) \leqslant \tilde{\epsilon}$. Lemma 4.2 applied to each diagonal block of δX_ℓ implies that there is a block diagonal matrix $\delta\tilde{X}_j$, with the same block structure as δX_ℓ, such that each diagonal block has rank at most $6kh$ and $\|\delta X_\ell - \delta\widetilde{X}_\ell\|_2 \leq \tilde{\epsilon}\|\delta X_\ell\|_2$. By construction, $\tilde{X} = X_0 + \delta\tilde{X}_0 + \cdots + \delta\tilde{X}_{p-1}$ is a $(\mathcal{T}_p, 6khp)$-HODLR matrix such that $\|X - \tilde{X}\|_2 \leq \tilde{\epsilon} p \max_\ell \|\delta X_\ell\|_2$. This establishes the following result.

COROLLARY 4.3. *Under the assumptions of Lemma 4.2, let X be the solution of (1) and suppose that the norm of all corrections computed by Algorithm 4 is bounded*

by M. Given $\epsilon > 0$, let h be the smallest integer that satisfies $(1+\sqrt{2})Z_h(E,F) \leqslant \frac{\epsilon}{pM}$. Then there exists a $(\mathcal{T}_p, 6khp)$-HODLR matrix $\widetilde{X}$ such that $\|X - \widetilde{X}\|_2 \leqslant \epsilon$.

Remark 4.4. To turn Corollary 4.3 into an asymptotic statement on the HODLR rank as $n \to \infty$, one needs to assume a model for the behavior of E, F as $n \to \infty$. In the simplest case, E, F stay constant, for example, when A and B are symmetric positive definite and their condition numbers remain bounded as $n \to \infty$. In this case, the integer h from Corollary 4.3 is constant and, in turn, the HODLR rank of the approximate solution is $\mathcal{O}(k\log(n))$. Numerical tests, involving random HODLR matrices which satisfy these assumptions, indicate that the factor $\log(n)$ is *in general* not avoidable.

In many cases of practical interest, A and B are symmetric positive definite but their condition numbers grow polynomially with respect to n. For example, the condition number of $A = B = \mathrm{trid}(-1, 2, -1)$ is $\mathcal{O}(n^2)$. Using the result of Corollary 2.2 one now has $h = \mathcal{O}(\log(n))$ and, in turn, Corollary 4.3 yields the HODLR rank $\mathcal{O}(k\log^2(n))$.

4.2.2. Complexity of divide-and-conquer in the HODLR format. The complexity of Algorithm 4 depends on the convergence of the extended Krylov subspace method used for solving the correction equation in line 10 and, in turn, on spectral properties of A, B; see [9, 39]. To give some insight into the complexity, we make the following simplifying assumptions:

(i) the conditions of Lemma 4.2 are satisfied for sets E, F independent of n, and Algorithm 2 converges in a constant number of iterations;
(ii) $n = 2^p s$ and the input matrices A, B, and C are $(\mathcal{T}_p, k)$-HODLR matrices;
(iii) $\mathcal{T}_p$ is the perfectly balanced binary tree of depth p,
(iv) the compression in line 11 of Algorithm 4 is *not* performed.

We recall that the LU decomposition of a $(\mathcal{T}_p, k)$-HODLR matrix requires $\mathcal{O}(k^2 n\log^2(n))$ operations, while multiplying a vector with such a matrix requires $\mathcal{O}(kn\log(n))$ operations; see, e.g., [34].

Now, let $\mathcal{C}(n,k)$ denote the complexity of Algorithm 4. Assumption (i) implies that the cost of Algorithm 10, called at line 10, is $\mathcal{O}(k^2 n\log^2(n))$, because it is dominated by the cost of precomputing the LU decompositions for A and B. According to Corollary 4.3 and Remark 4.4, assumption (i) also implies that X_0 (see line 8) has HODLR rank $\mathcal{O}(k\log(n))$. This, together with the fact that U_B and V_A each have $2k$ columns, shows that the matrix multiplications $X_0 U_B$ and $X_0^* V_A$ at line 9 require $\mathcal{O}(k^2 n\log^2(n))$ operations. Finally, observe that the solution of a dense Sylvester equation with $s \times s$ coefficients requires $\mathcal{O}(s^3)$ operations. In summary,

$$\mathcal{C}(n,k) = \begin{cases} \mathcal{O}(s^3) & \text{if } n = s, \\ \mathcal{O}(k^2 n\log^2(n)) + 2\mathcal{C}(\frac{n}{2}, k) & \text{otherwise.} \end{cases}$$

Applying the master theorem [19, Theorem 4.1] to this relation yields $\mathcal{C}(n,k) = \mathcal{O}(k^2 n\log^3(n))$.

4.3. HSS matrices. The storage of a matrix of HODLR rank k requires $\mathcal{O}(kn\log n)$ memory in the HODLR format. Under stronger conditions on the matrix, the factor $\log n$ can be avoided by using nested hierarchical low-rank formats, such as the HSS format.

An HSS matrix is partitioned in the same way as a HODLR matrix. By Definition 4.1, a matrix A is a $(\mathcal{T}_p, k)$-HODLR matrix if and only if every off-diagonal block

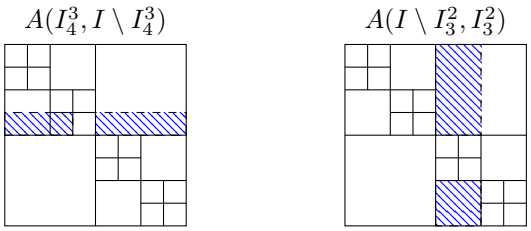

FIG. 4. *Illustration of an HSS block row and an HSS block column for a cluster tree of depth* 3.

$A(I_i^\ell, I_j^\ell)$, $i \neq j$, has rank at most k. Thus, we have a factorization

$$A(I_i^\ell, I_j^\ell) = U_i^{(\ell)} S_{i,j}^{(\ell)} (V_j^{(\ell)})^*, \quad S_{i,j}^{(\ell)} \in \mathbb{C}^{k\times k}, \quad U_i^{(\ell)} \in \mathbb{C}^{n_i^{(\ell)}\times k}, \quad V_j^{(\ell)} \in \mathbb{C}^{n_j^{(\ell)}\times k},$$

where we assume exact rank k to simplify the description. The crucial extra assumption for HSS matrices is that the bases matrices of these low-rank representations are nested across the different levels. That is, one assumes that there exist matrices, the so-called translation operators, $R_{U,i}^{(\ell)}, R_{V,j}^{(\ell)} \in \mathbb{C}^{2k\times k}$ such that

$$U_i^{(\ell)} = \begin{bmatrix} U_{2i-1}^{(\ell+1)} & 0 \\ 0 & U_{2i}^{(\ell+1)} \end{bmatrix} R_{U,i}^{(\ell)}, \qquad V_j^{(\ell)} = \begin{bmatrix} V_{2j-1}^{(\ell+1)} & 0 \\ 0 & V_{2j}^{(\ell+1)} \end{bmatrix} R_{V,j}^{(\ell)}. \tag{24}$$

This nestedness condition allows us to construct the bases $U_i^{(\ell)}$ and $V_i^{(\ell)}$ for any level $\ell = 1, \ldots, p-1$ recursively from the bases $U_i^{(p)}$ and $V_i^{(p)}$ at the deepest level p using the matrices $R_{U,i}^{(\ell)}$ and $R_{V,j}^{(\ell)}$. In turn, one only needs $O(nk)$ memory to represent A, for storing the diagonal blocks $A(I_i^p, I_i^p)$, the bases $U_i^{(p)}$, $V_i^{(p)}$ as well as $S_{2i-1,2i}^{(\ell)}$, $S_{2i,2i-1}^{(\ell)}$, $R_{U,i}^{(\ell)}$, $R_{V,i}^{(\ell)}$.

As explained in [59], the described construction is possible if and only if all the corresponding block rows and columns, without their diagonal blocks, have rank at most k on every level. The following definition formalizes and extends this condition.

DEFINITION 4.5. *Consider a cluster tree $\mathcal{T}_p$ and let $A \in \mathbb{C}^{n\times n}$. Then,*

(a) *$A(I_i^\ell, I \setminus I_i^\ell)$ is called an HSS block row and $A(I \setminus I_i^\ell, I_i^\ell)$ is called an HSS block column for $i = 1, \ldots, 2^\ell$, $\ell = 1, \ldots, p$;*
(b) *for $k \in \mathbb{N}$, A is called a $(\mathcal{T}_p, k)$-HSS matrix if every HSS block row and column of A has rank at most k;*
(c) *the HSS rank of A (with respect to $\mathcal{T}_p$) is the smallest integer k such that A is a $(\mathcal{T}_p, k)$-HSS matrix;*
(d) *for $k \in \mathbb{N}$ and $\epsilon > 0$, A is called an ϵ-$(\mathcal{T}_p, k)$-HSS matrix if every HSS block row and column of A admits an ϵ-approximation of rank k.*

An illustration of HSS block rows and columns is given in Figure 4.

By Definition 4.5(b), a matrix A with bandwidth b (i.e., $a_{ij} = 0$ for $|i-j| > b$) has HSS rank at most $2b$.

It is intuitive that a matrix satisfying Definition 4.5(d) can be approximated by a $(\mathcal{T}_p, k)$-HSS matrix with an error of norm proportional to ϵ. Such a result has been shown for the Frobenius norm in [58, Corollary 4.3]. In the following, we show such a result for the matrix 2-norm, with a constant that is tighter than what can be directly concluded from [58]. For this purpose, we first introduce some notation and

preliminary results. In the following, we say that a block diagonal matrix D conforms with $\mathcal{T}_p$ if it takes the form

$$D = D_1 \oplus D_2 \oplus \cdots \oplus D_{2^p}, \qquad D_i \in \mathbb{C}^{n_i^{(p)} \times k_i},$$

for integers $k_i \le n_i^{(p)}$. In particular, this ensures that the multiplications $D^T A$ and AD do not mix different HSS block rows and columns, respectively. In the following lemma, we let $\mathcal{T}_p^{(k)}$ denote the tree associated with the partition $k_1 + k_2 + \cdots + k_{2^p}$.

LEMMA 4.6. *Let A be an ϵ-$(\mathcal{T}_p, k)$-HSS matrix. Then,*

(a) *UU^TA and U^TAV are ϵ-$(\mathcal{T}_p, k)$-HSS and ϵ-$(\mathcal{T}_p^{(k)}, k)$-HSS matrices, respectively, for any block diagonal matrices U, V conforming with $\mathcal{T}_p$ and satisfying $U^TU = V^TV = I$;*

(b) *A is an ϵ-$(\mathcal{T}_{p-1}, k)$-HSS matrix for the tree $\mathcal{T}_{p-1}$ obtained from $\mathcal{T}_p$ by omitting all leaves.*

Proof. (a) Consider a node I_i^ℓ of $\mathcal{T}_p$ and the corresponding node $\tilde{I}_i^\ell$ of $\mathcal{T}_p^{(k)}$.

Because of the block diagonal structure of U, an HSS block row of UU^TA takes the form $\Pi A(I_i^\ell, I \setminus I_i^\ell)$, where $\Pi = U(I_i^\ell, \tilde{I}_i^\ell)U(I_i^\ell, \tilde{I}_i^\ell)^T$ is an orthogonal projector. By assumption, there is a perturbation C with $\|C\|_2 \le \epsilon$ such that $A(I_i^\ell, I \setminus I_i^\ell) + C$ has rank at most k. In turn, $\Pi A(I_i^\ell, I \setminus I_i^\ell) + \Pi C$ also has rank at most k with $\|\Pi C\|_2 \le \|\Pi\|_2 \|C\|_2 = \|C\|_2 \le \epsilon$. By an analogous argument, the HSS block column $U(I \setminus I_i^\ell, \tilde{I} \setminus \tilde{I}_i^\ell)U(I \setminus I_i^\ell, \tilde{I} \setminus \tilde{I}_i^\ell)^T A(I \setminus I_i^\ell, I_i^\ell)$ is shown to admit a rank-k approximation of norm ϵ. This proves that UU^TA is an ϵ-$(\mathcal{T}_p, k)$-HSS matrix.

Now, consider an HSS block row of U^TAV given by $U(I_i^\ell, \tilde{I}_i^\ell)^T A(I_i^\ell, I \setminus I_i^\ell)V(I \setminus I_i^\ell, \tilde{I} \setminus \tilde{I}_i^\ell)$, where both the left and right factors have orthonormal columns because of the structure of U, V. Using the matrix C from above, set

$$\tilde{C} := U(I_i^\ell, \tilde{I}_i^\ell)^T C(I \setminus I_i^\ell, \tilde{I} \setminus \tilde{I}_i^\ell)V(I \setminus I_i^\ell, \tilde{I} \setminus \tilde{I}_i^\ell).$$

This perturbation reduces the rank of the HSS block row to at most k and has norm bounded by ϵ because of the ortogonality of U, V. By swapping the roles of U and V, the same holds for an HSS block column of U^TAV. This proves that U^TAV is an ϵ-$(\mathcal{T}_p^{(k)}, k)$-HSS matrix.

(b) This part trivially holds because the block rows and columns corresponding to $\mathcal{T}_{p-1}$ are a subset of the block rows and columns corresponding to $\mathcal{T}_p$. ∎

THEOREM 4.7. *Let $A \in \mathbb{C}^{n \times n}$ be an ϵ-$(\mathcal{T}_p, k)$-HSS matrix. Then there exists $\delta A \in \mathbb{C}^{n \times n}$ with $\|\delta A\|_2 \le \sqrt{2^{p+2} - 4} \cdot \epsilon$ such that $A + \delta A$ is a $(\mathcal{T}_p, k)$-HSS matrix.*

Proof. The result is proven by induction on the tree depth p. The result trivially holds for $p = 0$.

Let us now consider $p \ge 1$ and suppose that the result holds for any ϵ-$(\mathcal{T}_{p-1}, k)$-HSS matrix and for any tree $\mathcal{T}_{p-1}$ of depth $p - 1$. To establish the result for a tree $\mathcal{T}_p$ of depth p, we consider the off-diagonal part A_{off}, that is, A_{off} is obtained from A by setting the diagonal blocks $A(I_i^p, I_i^p)$ to zero for $i = 1, \ldots, 2^p$. This allows us to consider the complete block rows $A_{\text{off}}(I_i^p, I) \in \mathbb{C}^{n_i^{(p)} \times n}$ instead of the depth-p HSS block rows of A. Let $U_i \in \mathbb{C}^{n_i^{(p)} \times k}$ contain the k left dominant singular vectors of $A_{\text{off}}(I_i^p, I)$ (if $k \le n_i^{(p)}$, we set $U_i = I_{n_i^{(p)}}$). Because A_{off} is an ϵ-$(\mathcal{T}_p, k)$-HSS matrix, it holds that $\|(I - U_iU_i^T)A_{\text{off}}(I_i^p, I)\|_2 \le \epsilon$. The block diagonal matrix $U := U_1 \oplus \cdots \oplus U_{2^p}$ conforms with $\mathcal{T}_p$ and it is such that

$$\|(I - UU^T)A_{\text{off}}\|_2 \le \sqrt{2^p}\epsilon. \tag{25}$$

By Lemma 4.6, $UU^T A_{\text{off}}$ is again an ϵ-$(\mathcal{T}_p, k)$-HSS matrix. This allows us to apply analogous arguments to the depth-p HSS block columns of $UU^T A_{\text{off}}$, yielding a block diagonal matrix V conforming with $\mathcal{T}_p$ such that $UU^T A_{\text{off}} VV^T$ has depth-p HSS block rows/columns of rank at most k, and

$$\|UU^T A_{\text{off}}\,(I - VV^T)\|_2 \leq \sqrt{2^p}\epsilon.$$

Using the notation from Lemma 4.6, $U^T A_{\text{off}} V$ is an ϵ-$(\mathcal{T}_p^{(k)}, k)$-HSS matrix and, in turn, an ϵ-$(\mathcal{T}_{p-1}^{(k)}, k)$-HSS matrix. We recall that $\mathcal{T}_{p-1}^{(k)}$ is the tree of depth $p-1$ obtained by eliminating the leaves of $\mathcal{T}_p^{(k)}$. Hence, the induction hypothesis implies the existence of $\delta_{p-1}A$ such that $U^T A_{\text{off}} V + \delta_{p-1}A$ is a $(\mathcal{T}_{p-1}^{(k)}, k)$-HSS matrix and

$$\|\delta_{p-1}A\|_2 \leq \sqrt{2^{p+1} - 4}\epsilon. \tag{26}$$

The matrix $UU^T A_{\text{off}} VV^T + U\delta_{p-1}AV^T$ is not only a $(\mathcal{T}_{p-1}, k)$-HSS matrix but also a $(\mathcal{T}_p, k)$-HSS matrix because, by construction, its depth-p HSS block rows and columns all have rank at most k. In summary, $A + \delta A$ is a $(\mathcal{T}_p, k)$ matrix with

$$\delta A := -(I - UU^T)A_{\text{off}} - UU^T A_{\text{off}}(I - VV^T) + U\delta_{p-1}A\,V^T.$$

Exploiting the orthogonality of (co-)ranges and using (25)–(26), the norm of this perturbation satisfies

$$\begin{aligned}\|\delta A\|_2^2 &\leq \|(I - UU^T)A_{\text{off}}\|_2^2 + \|UU^T A_{\text{off}}(I - VV^T) + U\delta_{p-1}A\,V^T\|_2^2 \\ &\leq \|(I - UU^T)A_{\text{off}}\|_2^2 + \|UU^T A_{\text{off}}(I - VV^T)\|_2^2 + \|U\delta_{p-1}A\,V^T\|_2^2 \\ &\leq 2^p\epsilon^2 + 2^p\epsilon^2 + (2^{p+1} - 4)\epsilon^2 = (2^{p+2} - 4)\epsilon^2,\end{aligned}$$

which completes the proof. ∎

4.4. Compressibility of solution X in the HSS format. We now consider the equation $AX + XB = C$ for $(\mathcal{T}_p, k)$-HSS coefficients A, B, C. Algorithm 4 can be adapted to this situation by simply replacing operations in the HODLR format by operations in the HSS format. In our numerical tests, the HSS compression algorithm from [59] is used in line 11. Moreover, sparse LU factorizations of the matrices A and B are obtained with the MATLAB function `lu`, in Algorithm 2. When A and B are nonsparse HSS matrices, one can use the algorithms described in [59] for precomputing either their ULV or Cholesky factorization.

In the following, we show that the solution X can be well approximated by an HSS matrix.

LEMMA 4.8. *Let $A, B, C \in \mathbb{C}^{n\times n}$ be $(\mathcal{T}_p, k)$-HSS matrices and suppose that $\mathcal{W}(A) \subseteq E$ and $\mathcal{W}(-B) \subseteq F$ for sets $E, F \subset \mathbb{C}$ satisfying $E \cap F = \emptyset$. Let Y be an HSS block row or column of the solution X of (1). Then*

$$\frac{\sigma_{3kh+1}(Y)}{\|Y\|_2} \leqslant \left(1 + \sqrt{2}\right) \cdot Z_h(E, F).$$

Proof. We establish the result for an HSS block column $X(I \setminus I_i^\ell, I_i^\ell)$; the case of an HSS block row is treated analogously. Our proof follows the proof of Theorem 2.7 in [47].

Let us define $A_{11} = A(I_i^\ell, I_i^\ell)$, $A_{21} = A(I \setminus I_i^\ell, I_i^\ell)$, $A_{12} = A(I_i^\ell, I \setminus I_i^\ell)$, $A_{22} = A(I \setminus I_i^\ell, I \setminus I_i^\ell)$, and B_{ij}, C_{ij}, X_{ij} analogously for $1 \leqslant i, j \leqslant 2$. Extracting the indices $(I \setminus I_i^\ell, I_i^\ell)$ from the equation $AX + XB = C$, we obtain the relation

$$A_{21}X_{11} + A_{22}X_{21} + X_{21}B_{11} + X_{22}B_{21} = C_{21}.$$

This shows that X_{21} satisfies a Sylvester equation with right-hand side of rank at most $3k$:

$$A_{22}X_{21} + X_{21}B_{11} = C_{21} - A_{21}X_{11} - X_{22}B_{21}.$$

Since $\mathcal{W}(A_{22}) \subseteq \mathcal{W}(A)$ and $\mathcal{W}(-B_{11}) \subseteq \mathcal{W}(-B)$, and $X(I \setminus I_i^\ell, I_i^\ell) = X_{21}$, the claim follows from Theorem 2.1. ∎

Combining Lemma 4.8 with Theorem 4.7 yields the following result.

COROLLARY 4.9. *Under the assumptions of Lemma* 4.8*, let X be the solution of* (1)*. Given $\epsilon > 0$, let h be the smallest integer that satisfies $(1+\sqrt{2})Z_h(E,F) \leqslant \frac{\epsilon}{\sqrt{2^{p+2}-4}}$. Then there exists a $(\mathcal{T}_p, 3kh)$-HSS matrix $\widetilde{X}$ such that $\|X - \widetilde{X}\|_2 \leqslant \epsilon$.*

4.4.1. Complexity of divide-and-conquer in the HSS format. The complexity analysis of Algorithm 4 from section 4.2.2 extends to the HSS format as follows. We retain assumptions (i) and (iii) from section 4.2.2 and replace (ii) and (iv) by the following:

(ii′) $n = 2^p s$ and the input matrices A, B, and C are $(\mathcal{T}_p, k)$-HSS matrices of size $n \times n$;

(iv′) the compression in line 11 of Algorithm 4 is performed and returns HSS rank $\mathcal{O}(k)$.

The second part of assumption (iv′) is motivated by the fact that the (exact) matrix $X_0 + \delta X$ is the solution of a Sylvester equation with the coefficients satisfying the conditions of Corollary 4.9. Applied recursively, assumption (iv′) implies that X_{11} and X_{22} have HSS rank $\mathcal{O}(k)$. Using the fact that matrix-vector multiplications with these matrices have complexity $\mathcal{O}(kn)$, line 9 requires $\mathcal{O}(k^2 n)$ operations. The LU factorizations of A and B needed in line 10 and the compression in line 11 have the same complexity [59]. Hence, by recursion, the overall complexity of Algorithm 4 in the HSS format is $\mathcal{O}(k^2 n \log(n))$.

4.4.2. Reducing the rank of updates in the Hermitian case. The splitting (21), the basis of our divide-and-conquer method, leads to perturbations of rank $2k$ for general $(\mathcal{T}_p, k)$-HODLR and HSS matrices. For a Hermitian positive definite $(\mathcal{T}_p, k)$-HSS matrix A, let $A_{21} = U\Sigma V^*$ be the SVD of the subdiagonal block on level 1. Instead of (21) we then consider the splitting

(27)
$$A = \begin{bmatrix} A_{11} & V\Sigma U^* \\ U\Sigma V^* & A_{22} \end{bmatrix} = A_0 + \delta A := \begin{bmatrix} A_{11} + V\Sigma V^* & 0 \\ 0 & A_{22} + U\Sigma U^* \end{bmatrix} + \begin{bmatrix} V \\ -U \end{bmatrix} \Sigma \begin{bmatrix} -V \\ U \end{bmatrix}^*.$$

The obvious advantage is that the perturbation now has rank k. However, in order to be a useful basis for the divide-and-conquer method, A_0 needs to inherit the favorable properties of A. This is shown by the following lemma.

LEMMA 4.10. *Let A be a Hermitian positive definite $(\mathcal{T}_p, k)$-HSS matrix, partitioned as in* (27)*. Then A_0 is also a Hermitian positive definite $(\mathcal{T}_p, k)$-HSS matrix.*

Proof. Note that

$$A_{11} + V\Sigma V^* = \begin{bmatrix} A_{11} & V\Sigma U^* \end{bmatrix} \begin{bmatrix} I \\ UV^* \end{bmatrix} = \begin{bmatrix} I & VU^* \end{bmatrix} \begin{bmatrix} A_{11} \\ U\Sigma V^* \end{bmatrix}.$$

The first relation implies that the rank of an HSS block row of $A_{11}+V\Sigma V^*$ is bounded by the rank of the corresponding HSS block row of $\begin{bmatrix} A_{11} & V\Sigma U^* \end{bmatrix}$, which is bounded by k. The second relation implies that the rank of an HSS block column of $A_{11}+V\Sigma V^*$ is bounded by the rank of the corresponding HSS block column of $\begin{bmatrix} A_{11} \\ U\Sigma V^* \end{bmatrix}$, which is also bounded by k. An analogous argument applies to the HSS block rows and columns of $A_{22}+U\Sigma U^*$. Thus, A_0 is a $(\mathcal{T}_p, k)$-HSS matrix. It is straightforward to verify that A_0 is Hermitian positive definite. ∎

The right-hand side $C = C_0 + \delta C$ in (1) is treated similarly, for lowering the rank of the right-hand side of (4). Since we do not need to preserve any positive definiteness in C_0, we are allowed to choose

$$\delta C = \begin{bmatrix} \theta V \\ -U \end{bmatrix} \Sigma \begin{bmatrix} -V \\ \theta^{-1} U \end{bmatrix}^* \text{ for } \theta \neq 0.$$

Remark 4.11. In the special case when A is a Hermitian banded matrix, with bandwidth smaller than s, the updates VV^* and UU^* only affect the smallest diagonal blocks in the southeast corner of A_{11} and in the northwest corner of A_{22}, respectively. In particular, the sparsity pattern of the off-diagonal blocks is maintained.

5. Numerical results. In this section, we illustrate the performance of our divide-and-conquer method from section 4 for a number of different examples. In particular, we consider linear matrix equations $AX + XB = C$ for which A, B, C are efficiently representable as HSS or HODLR matrices. We exclude cases where C has low rank (or low numerical rank), since these can be treated more efficiently directly by ADI or Krylov subspace methods.

We compare our method with other techniques developed for linear matrix equations with rank-structured coefficients. In particular, this includes the matrix sign function iteration for HODLR matrices proposed in [30] and recently tested in [47]. When the coefficients A and B are symmetric positive definite, well conditioned, and banded, we also compare with the approach proposed in [48], a matrix version of the CG that exploits the approximate sparsity in the solution.

A number of approaches are based on applying numerical quadrature to $X = \int_0^\infty e^{-tA} C e^{-tB} dt$, for example, in [33] in the context of sparsity and in [47] in the context of HODLR matrices. As demonstrated in [47] such approaches are less competitive compared to the sign iteration and they are therefore not included in our comparison.

5.1. Details of the implementation. All experiments have been performed on a laptop with a dual-core Intel Core i7-7500U 2.70 GHz CPU, 256 KB of level 2 cache, and 16 GB of RAM. The algorithms are implemented in MATLAB and tested under MATLAB 2016a, with MKL BLAS version 11.2.3 utilizing both cores.

The methods described in sections 3 and 4 require the choice of several parameters:

- τ_{NW} = tolerance for stopping the Newton method (see line 11 of Algorithm 3);
- τ_{EK} = tolerance for stopping the extended Krylov subspace method, (see (11));
- s = size of the diagonal blocks in the HODLR/HSS block partitioning;
- τ_σ = tolerance for low-rank truncations when compressing the right-hand side (see section 3.1), the output of Algorithm 2, as well as HODLR and HSS matrices in line 11 of Algorithm 4.

Concerning the compression in the HODLR and HSS formats, we specify that in each off-diagonal block we discard the singular values that are relatively small with respect

TABLE 3
Choices of parameters used in the experiments.

Test	τ_{NW}	τ_{EK}	s	τ_σ
Example 3.1	10^{-8}	10^{-8}	-	-
Example 3.2	10^{-8}	10^{-12}	-	-
Section 5.2	-	10^{-12}	256	10^{-12}
Section 5.3	-	10^{-12}	256	10^{-12}
Section 5.4	-	10^{-6}	256	10^{-6}
Section 5.5	10^{-8}	10^{-12}	256	10^{-12}
Section 5.6	10^{-8}	10^{-12}	256	10^{-12}

to the norm of the whole matrix, which can be cheaply estimated with a few steps of the power method.

The values of the parameters used in the various experiments are reported in Table 3. We have found that the performance of the proposed algorithms is not very sensitive to the choices of the tolerances reported in Table 3: smaller tolerances lead to more accurate results, as one would expect. It is, however, advisable to choose τ_{EK} and τ_σ on a similar level, in order to avoid wasting computational resources. The tolerance τ_{NW} can be chosen larger because the quadratic convergence of the Newton method implies that the actual error $\|X_{k+1} - X^*\|$ is proportional to τ_{NW}^2.

To assess the accuracy of an approximate solution $\hat{X}$ of a Sylvester equation, we report the residual $\mathrm{Res}(\hat{X}) = \|A\hat{X} + \hat{X}B - C\|_2/((\|A\|_2 + \|B\|_2)\|\hat{X}\|_2)$ which is linked to the relative backward error on the associated linear system [36]. For CAREs we consider the quantity $\mathrm{Res}(\hat{X}) = \|A\hat{X} + \hat{X}A^* - \hat{X}B\hat{X} - C\|_2/\|AX_0 + X_0A^* - X_0BX_0 - C\|_2$ instead.

5.2. Discretized two-dimensional Laplace equation. We consider the two-dimensional Laplace equation

$$\begin{cases} -\Delta u = f(x,y), & (x,y) \in \Omega, \\ u(x,y) = 0, & (x,y) \in \partial\Omega, \end{cases} \qquad \Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \tag{28}$$

for the square domain $\Omega = [0,1]^2$ and $f(x,y) = \log(1 + |x - y|)$. It is well known that the central finite difference discretization (28) on a regular grid leads to a Lyapunov equation $AX + XA = C$ with coefficients $A = (n+1)^2 \cdot \mathrm{trid}(-1, 2, -1)$ and C containing samples of $f(x,y)$ on the grid. The latter matrix does not have low (numerical) rank, but it can be well approximated in the HODLR and HSS formats relying on the Chebyshev expansion of f in the off-diagonal subdomains of Ω; see the discussion in [47, Example 6.1].

Table 4 and Figure 5 compare the performance of the matrix sign function iteration in the HODLR format with the divide-and-conquer method in both the HODLR and HSS formats.

The divide-and-conquer method is based on extended Krylov subspaces with principal submatrices of A. As these matrices inherit the tridiagonal structure of A, they can be easily applied and inverted. In contrast, the matrix sign iteration method does not preserve bandedness and needs to operate with general HODLR matrices. This is significantly less efficient; in turn, our divide-and-conquer method is always faster and scales more favorably as n increases. Moving from the HODLR to the HSS format results in further (modest) speedup. The HODLR and HSS ranks remain reasonably small in all approaches. One major advantage of the HSS format is its reduced mem-

TABLE 4
Execution times (in seconds) and relative residuals for the matrix sign function iteration and the divide-and-conquer method applied to the discretized two-dimensional Laplace equation from section 5.2.

n	T_{Sign}	Res_{Sign}	$T_{D\&C\ HODLR}$	$Res_{D\&C\ HODLR}$	$T_{D\&C\ HSS}$	$Res_{D\&C\ HSS}$
512	0.57	$9.04 \cdot 10^{-13}$	0.42	$4.32 \cdot 10^{-13}$	0.32	$6.71 \cdot 10^{-13}$
1,024	1.85	$1.60 \cdot 10^{-12}$	1.16	$7.70 \cdot 10^{-13}$	1.38	$7.36 \cdot 10^{-13}$
2,048	5.45	$2.09 \cdot 10^{-12}$	3.19	$7.51 \cdot 10^{-13}$	3.33	$9.86 \cdot 10^{-13}$
4,096	15.22	$3.39 \cdot 10^{-12}$	7.45	$6.85 \cdot 10^{-13}$	8.32	$8.03 \cdot 10^{-13}$
8,192	40.48	$5.69 \cdot 10^{-12}$	16.89	$8.01 \cdot 10^{-13}$	16.85	$7.47 \cdot 10^{-13}$
16,384	97.3	$7.04 \cdot 10^{-12}$	39.87	$6.84 \cdot 10^{-13}$	38.59	$7.37 \cdot 10^{-13}$
32,768	242.96	$8.94 \cdot 10^{-12}$	86.28	$7.08 \cdot 10^{-13}$	88.27	$8.89 \cdot 10^{-13}$
65,536	591.21	$1.01 \cdot 10^{-11}$	189.73	$6.45 \cdot 10^{-13}$	184.94	$9.64 \cdot 10^{-13}$
$1.31 \cdot 10^5$	1,433.9	$9.99 \cdot 10^{-12}$	393.95	$7.10 \cdot 10^{-13}$	377.44	$1.06 \cdot 10^{-12}$

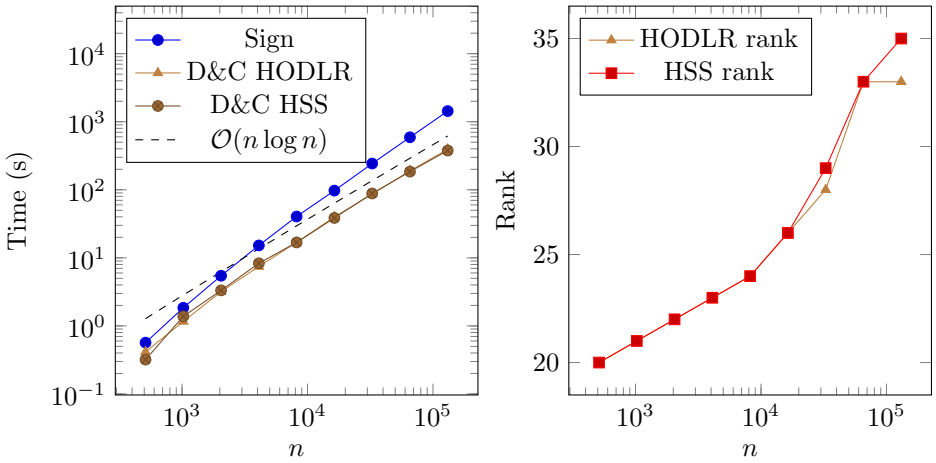

FIG. 5. *On the left, timings of the algorithms applied to the Laplacian equation with respect to the grid size. The dashed lines report the expected theoretical complexity of the divide-and-conquer strategies. On the right, HODLR and HSS rank of the solutions returned by the divide-and-conquer methods.*

ory requirements; for example, for $n = 1.31 \cdot 10^5$, 433 MB and 267 MB are required to store the approximate solution in the HODLR and HSS formats, respectively.

5.3. Convection diffusion. We repeat the experiment from section 5.2 for the convection-diffusion equation

$$\begin{cases} -\Delta u + v\nabla u = f(x,y), & (x,y) \in \Omega := [0,1], \\ u(x,y) = 0, & (x,y) \in \partial\Omega, \end{cases}$$

where $v = [10, 10]$ and, once again, $f(x,y) = \log(1 + |x - y|)$. A standard finite difference discretization now leads to a Lyapunov equation $AX + XA^T = C$ with the nonsymmetric matrix

$$A = (n+1)^2 \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} + \frac{5}{2}(n+1) \begin{bmatrix} 3 & -5 & 1 & & \\ 1 & 3 & -5 & \ddots & \\ & \ddots & \ddots & \ddots & 1 \\ & & 1 & 3 & -5 \\ & & & 1 & 3 \end{bmatrix}$$

TABLE 5

Execution times (in seconds) and relative residuals for the matrix sign function iteration and the divide-and-conquer method applied to the discretized convection-diffusion equation from section 5.3.

n	$\mathrm{T}_{\mathrm{Sign}}$	$\mathrm{Res}_{\mathrm{Sign}}$	$\mathrm{T}_{\mathrm{D\&C\ HODLR}}$	$\mathrm{Res}_{\mathrm{D\&C\ HODLR}}$	$\mathrm{T}_{\mathrm{D\&C\ HSS}}$	$\mathrm{Res}_{\mathrm{D\&C\ HSS}}$
512	0.6	$1.15 \cdot 10^{-12}$	0.42	$4.85 \cdot 10^{-13}$	0.38	$4.85 \cdot 10^{-13}$
1,024	1.87	$9.51 \cdot 10^{-13}$	1.47	$6.59 \cdot 10^{-13}$	1.47	$7.66 \cdot 10^{-13}$
2,048	5.42	$1.78 \cdot 10^{-12}$	3.22	$4.51 \cdot 10^{-13}$	3.48	$5.57 \cdot 10^{-13}$
4,096	15.82	$2.94 \cdot 10^{-12}$	8.37	$4.62 \cdot 10^{-13}$	8.08	$7.64 \cdot 10^{-13}$
8,192	40.15	$4.38 \cdot 10^{-12}$	21.48	$7.56 \cdot 10^{-13}$	19.64	$6.47 \cdot 10^{-13}$
16,384	99.18	$6.52 \cdot 10^{-12}$	40.41	$6.23 \cdot 10^{-13}$	40.99	$8.83 \cdot 10^{-13}$
32,768	236.81	$8.12 \cdot 10^{-12}$	94.9	$8.30 \cdot 10^{-13}$	89.72	$6.96 \cdot 10^{-13}$
65,536	603.59	$8.37 \cdot 10^{-12}$	198.04	$8.63 \cdot 10^{-13}$	207.62	$8.06 \cdot 10^{-13}$
$1.31 \cdot 10^5$	1,365.6	$7.85 \cdot 10^{-12}$	430.47	$8.52 \cdot 10^{-13}$	418.08	$8.43 \cdot 10^{-13}$

TABLE 6

Execution times (in seconds) and relative residuals for the sparse CG method and the divide-and-conquer method applied to the head equation from section 5.4.

n	T_{CG}	$\mathrm{Res}_{\mathrm{CG}}$	$\mathrm{T}_{\mathrm{D\&C\ HODLR}}$	$\mathrm{Res}_{\mathrm{D\&C\ HODLR}}$	$\mathrm{T}_{\mathrm{D\&C\ HSS}}$	$\mathrm{Res}_{\mathrm{D\&C\ HSS}}$
1,536	0.86	$5.95 \cdot 10^{-8}$	0.86	$1.23 \cdot 10^{-8}$	0.76	$1.23 \cdot 10^{-8}$
3,072	1.77	$5.86 \cdot 10^{-8}$	2.03	$1.24 \cdot 10^{-8}$	2.5	$1.23 \cdot 10^{-8}$
6,144	4.39	$5.76 \cdot 10^{-8}$	4.62	$1.24 \cdot 10^{-8}$	4.61	$1.23 \cdot 10^{-8}$
12,288	8.08	$5.71 \cdot 10^{-8}$	11.79	$1.24 \cdot 10^{-8}$	10.98	$1.23 \cdot 10^{-8}$
24,576	17.06	$5.68 \cdot 10^{-8}$	22.72	$1.23 \cdot 10^{-8}$	24.61	$1.23 \cdot 10^{-8}$
49,152	35.55	$5.67 \cdot 10^{-8}$	58.24	$1.23 \cdot 10^{-8}$	53.1	$1.23 \cdot 10^{-8}$
98,304	71.13	$5.66 \cdot 10^{-8}$	128.28	$1.23 \cdot 10^{-8}$	125.32	$1.23 \cdot 10^{-8}$

and C as in section 5.2. Table 5 displays the timings and the relative residuals obtained for this example, reconfirming our observations for the symmetric example from section 5.2. Also, we have observed the HODLR and HSS ranks to behave in a similar manner.

5.4. Heat equation. A model describing the temperature change of a thermally actuated deformable mirror used in extreme ultraviolet litography [33, section 5.1] leads to a symmetric Lyapunov equation $AX + XA = C$ with coefficients

$$A = I_q \otimes \mathrm{trid}_6(b, a, b) + \mathrm{trid}_q(b, 0, b) \otimes I_6,$$
$$C = I_q \otimes (-c \cdot E_6 + (c-1) \cdot I_6) + \mathrm{trid}_q(d, 0, d) \otimes E_6,$$

where $a = -1.36$, $b = 0.34$, $c = 0.2$, $d = 0.1$ and E_6 is the 6×6 matrix of all ones. Note that A and C are banded with bandwidth 6 and 11, respectively. As analyzed in [48, Ex. 2.6], the condition number of A is bounded by 40 and hence the sparse CG method proposed there scales linearly with $n := 6q$. We executed the sparse CG setting $X_0 := 0$ and using $\|AX_k + X_k A - C\|_F / \|C\|_F \leqslant 10^{-6}$, for stopping the iterations, as suggested in [48]. The results are reported in Table 6. Although it is faster and its advantageous scaling is clearly visible, approximate sparsity is, compared to the HODLR and HSS formats significantly less effective at compressing the solution X; see Figure 6. The observed HODLR and HSS ranks are equal to 10 and 20, respectively, independently of n.

5.5. A large-scale Riccati equation with banded and low-rank coefficients. In this example, we demonstrate how a combination of Algorithms 3 and 4 can be used to address certain large-scale Riccati equations. Consider the CARE

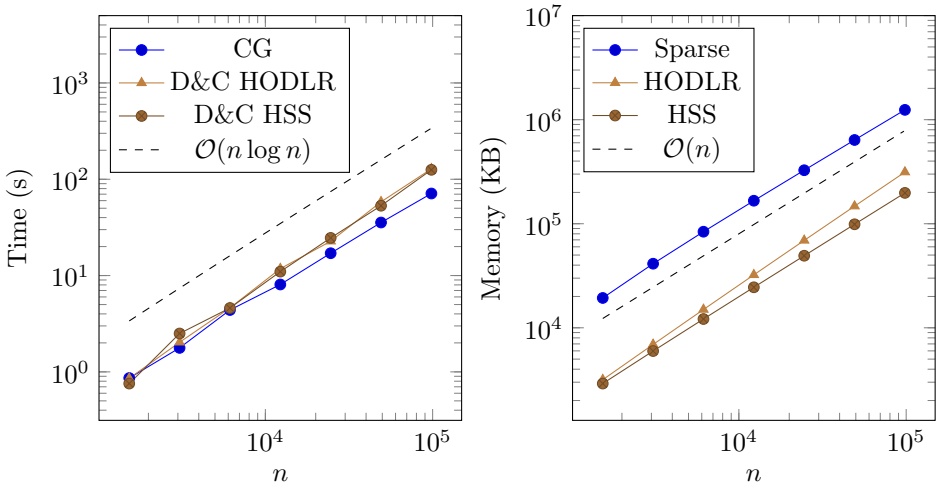

FIG. 6. *Time (left) and memory (right) consumptions for solving the heat equation with different grid sizes. The dashed lines report the expected asymptotic complexity for the divide-and-conquer with HSS matrices.*

TABLE 7

Performance of Algorithm 3*, combined with Algorithm* 4 *in the HSS format for the first step, for the CARE from section* 5.5.

n	$\|\hat{X}\|_2$	$\mathrm{T_{tot}}$	$\frac{T_{\text{step 1}}}{T_{\text{tot}}}$	$\mathrm{T_{avg}}$	Res	it	HSS rank
1,024	$3.11 \cdot 10^4$	0.79	0.36	0.06	$2.58 \cdot 10^{-7}$	9	14
2,048	$1.24 \cdot 10^5$	1.63	0.35	0.12	$1.29 \cdot 10^{-6}$	10	16
4,096	$4.96 \cdot 10^5$	3.57	0.34	0.26	$6.55 \cdot 10^{-6}$	10	19
8,192	$1.98 \cdot 10^6$	8.34	0.31	0.58	$3.10 \cdot 10^{-5}$	11	21
16,384	$7.93 \cdot 10^6$	21.39	0.29	1.53	$1.10 \cdot 10^{-4}$	11	24
32,768	$3.17 \cdot 10^7$	55.53	0.21	3.97	$5.32 \cdot 10^{-4}$	12	25
65,536	$1.27 \cdot 10^8$	170.61	0.16	13.07	$2.82 \cdot 10^{-3}$	12	29
$1.31 \cdot 10^5$	$5.08 \cdot 10^8$	475.11	0.11	35.07	$2.36 \cdot 10^{-2}$	13	31

$AX + XA^* - XBX - C = 0$ with the coefficients

$$A = \mathrm{trid}_n(1, -2, 1) \in \mathbb{R}^{n \times n}, \quad B = B_U B_U^T, \quad B_U = \begin{bmatrix} e_1 & e_n \end{bmatrix} \in \mathbb{R}^{n \times 2}, \quad C = -I_n.$$

As the matrix A is negative definite, we can choose $X_0 = 0$ as the stabilizing initial guess in the Newton method. In turn, the Lyapunov equation in the first step (see line 4 of Algorithm 3) takes the form $AX + XA = C$. Exploiting the structure of the coefficients, we address this equation with Algorithm 4 in the HSS format. For all subsequent iterations, we use the low-rank update procedure described in section 3.5, recompressing the intermediate solution X_k in the HSS format.

In contrast to the observations made in section 3.5.1, the results displayed in Table 7 now reveal that the first step does not dominate the cost of the overall algorithm. Note that T_{avg}, the average time per Newton step, grows more than linearly as n increases, due to the fact that the condition number of A increases and, in turn, the extended Krylov subspace method converges more slowly. As n increases, the norm of the final solution grows accordingly to the final residue. The HSS rank of the approximate solution X grows slowly, apparently only logarithmically with n.

5.6. A large-scale Riccati equation from a second-order problem. Let us consider a linear second-order control system

$$M\ddot{z} + L\dot{z} = Kz = Du,$$

where M is diagonal and K, L are banded (or, more generally, HSS). Applying linear-optimal control leads, after a suitable linearization, to a CARE (14) with the matrix A taking the form[1]

$$A = \begin{bmatrix} 0 & -M^{-1}K \\ I_q & -M^{-1}L \end{bmatrix}. \tag{29}$$

In fact, the matrix A from Example 3.2 is of this type, with K tridiagonal and M, L (scaled) identity matrices. It turns out that A does *not* have low HODLR or HSS rank. In the following, we explain a simple trick to turn A into an HSS matrix, which then allows us to apply and the techniques from section 5.5 to Example 3.2.

We first observe that the matrix A from (29) can be decomposed as

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \otimes I_q - \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \otimes M^{-1}K - \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes M^{-1}L.$$

Let Π denote the perfect shuffle permutation [57], which swaps the order in the Kronecker product of matrices of sizes 2 and q: $\Pi(X \otimes Y)\Pi^* = Y \otimes X$, for any $X \in \mathbb{C}^{2\times 2}$, $Y \in \mathbb{C}^{q\times q}$. Hence,

$$\widetilde{A} := \Pi A \Pi^* = I_q \otimes \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} - M^{-1}K \otimes \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - M^{-1}L \otimes \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}. \tag{30}$$

The following result allows us to control the HSS ranks for each of the terms.

LEMMA 5.1. *Let $A \in \mathbb{C}^{q\times q}$ be an $(\mathcal{T}_p, k_A)$-HSS matrix, and let $B \in \mathbb{C}^{m\times m}$ have rank k_B. Then $A \otimes B$ is a $(\mathcal{T}_p^{(mq)}, k_A k_B)$-HSS matrix, where $\mathcal{T}_p^{(mq)}$ is the cluster tree defined by the integer partition*

$$mq = mq_1 + mq_2 + \cdots + mq_{2^p}.$$

Proof. The results follows immediately considering that an HSS block row $\hat{X}$ in A corresponds to the HSS block row $\hat{X} \otimes B$ in $A \otimes B$, with respect to $\mathcal{T}_p^{(m)}$. If the former has rank bounded by k_A, then the latter has rank bounded by $k_A k_B$. Analogously for HSS block columns. ∎

Lemma 5.1 implies that the matrix $\widetilde{A}$ from (30) is a $(\mathcal{T}_p^{(2)}, k_1 + k_2)$-HSS if $M^{-1}K$ and $M^{-1}L$ are $(\mathcal{T}_p, k_1)$- and $(\mathcal{T}_p, k_2)$-HSS matrices, respectively. For Example 3.2 these assumptions are satisfied with $k_1 = 0, k_2 = 1$. In turn this allows us to apply the techniques from section 5.5 to the shuffled Riccati equation from Example 3.2, using the shuffled starting guess $\Pi X_0 \Pi^*$. Table 8 displays the obtained results. We

[1] Note that, in contrast to [1], we use the second companion linearization in order to be consistent with our choice of transposes in (14).

TABLE 8
Performance of Algorithm 3*, combined with Algorithm* 4 *in the HSS format for the first step, for the (shuffled) CARE from section* 5.6.

n	$\|\hat{X}\|_2$	$\mathrm{T_{tot}}$	$\frac{T_{\text{step 1}}}{\mathrm{T_{tot}}}$	$\mathrm{T_{avg}}$	Res	it	HSS rank
512	$1.55 \cdot 10^4$	0.88	0.52	0.05	$2.90 \cdot 10^{-8}$	9	13
1,024	$6.19 \cdot 10^4$	1.14	0.4	0.08	$1.34 \cdot 10^{-7}$	10	13
2,048	$2.48 \cdot 10^5$	2	0.39	0.14	$6.44 \cdot 10^{-7}$	10	16
4,096	$9.91 \cdot 10^5$	4.5	0.36	0.29	$2.41 \cdot 10^{-6}$	11	19
8,192	$3.96 \cdot 10^6$	10.01	0.33	0.67	$1.00 \cdot 10^{-5}$	11	20
16,384	$1.59 \cdot 10^7$	23.82	0.29	1.54	$7.33 \cdot 10^{-5}$	12	23
32,768	$6.34 \cdot 10^7$	60.67	0.24	4.19	$3.36 \cdot 10^{-4}$	12	24

highlight that the nonsymmetric Lyapunov equation that is solved in the first step of the Newton method does not satisfy the hypotheses of Lemma 4.2. In fact, the field of values of the matrix $A - X_0 B$ is not contained in the open left half plane. Still, Algorithm 4 is observed to perform very well.

6. Concluding remarks. We have proposed a Krylov subspace method for updating the solution of linear matrix equations whose coefficients are affected by low-rank perturbations. We have shown that our approach can significantly speed up the Newton iteration for solving certain CAREs. Moreover, we have designed a divide-and-conquer algorithm for linear matrix equations with hierarchically low-rank coefficients. A theoretical analysis of the structure preservation and of the computational cost has been provided. In the numerical tests, we have verified that our algorithm scales well with the size of the problem and often outperforms existing techniques that rely on approximate sparsity and data sparsity.

During this work, we encountered two issues that might deserve further investigation: (1) The structure of a stable Lyapunov equation is currently exploited only partially; see section 3.4. In particular, it is an open problem to design a divide-and-conquer method that aims directly at the Cholesky factor of the solution and thus preserves its semidefiniteness. (2) As seen in section 5.4, it can be advantageous to exploit (approximate) sparsity in the case of well-conditioned equations. It would be interesting to design a variant of the divide-and-conquer method that benefits from sparsity as well.

Acknowledgments. We thank Jianlin Xia for pointing out [60] and thank the referees for their careful reading and helpful remarks.

REFERENCES

[1] J. ABELS AND P. BENNER, *CAREX—A Collection of Benchmark Examples for Continuous-Time Algebraic Riccati Equations* (Version 2.0), SLICOT working note 1999-14, 1999, http://www.slicot.org.

[2] S. AMBIKASARAN AND E. DARVE, *An $\mathcal{O}(N \log N)$ fast direct solver for partial hierarchically semi-separable matrices: With application to radial basis function interpolation*, J. Sci. Comput., 57 (2013), pp. 477–501, https://doi.org/10.1007/s10915-013-9714-z.

[3] A. C. ANTOULAS, *Approximation of Large-Scale Dynamical Systems*, SIAM, Philadelphia, 2005.

[4] A. C. ANTOULAS, D. C. SORENSEN, AND Y. ZHOU, *On the decay rate of Hankel singular values and related issues*, Systems Control Lett., 46 (2002), pp. 323–342, https://doi.org/10.1016/S0167-6911(02)00147-0.

[5] J. BAKER, M. EMBREE, AND J. SABINO, *Fast singular value decay for Lyapunov solutions with nonnormal coefficients*, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 656–668, https://doi.org/10.1137/140993867.

[6] J. Ballani and D. Kressner, *Matrices with hierarchical low-rank structures*, in Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications, Lecture Notes in Math. 2173, Springer, New York, 2016, pp. 161–209.

[7] R. H. Bartels and G. W. Stewart, *Algorithm* 432: *The solution of the matrix equation* $AX + XB = C$, Commun. ACM, 15 (1972), pp. 820–826, http://doi.org/10.1145/361573.361582.

[8] U. Baur and P. Benner, *Factorized solution of Lyapunov equations based on hierarchical matrix arithmetic*, Computing, 78 (2006), pp. 211–234, http://doi.org/10.1007/s00607-006-0178-y.

[9] B. Beckermann, *An error analysis for rational Galerkin projection applied to the Sylvester equation*, SIAM J. Numer. Anal., 49 (2011), pp. 2430–2450, https://doi.org/10.1137/110824590.

[10] B. Beckermann and A. Townsend, *On the singular values of matrices with displacement structure*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1227–1248, https://doi.org/10.1137/16M1096426.

[11] P. Benner, P. Ezzatti, D. Kressner, E. S. Quintana-Ortí, and A. Remón, *A mixed-precision algorithm for the solution of Lyapunov equations on hybrid CPU-GPU platforms*, Parallel Comput., 37 (2011), pp. 439–450, https://doi.org/10.1016/j.parco.2010.12.002.

[12] P. Benner, P. Kürschner, and J. Saak, *Self-generating and efficient shift parameters in ADI methods for large Lyapunov and Sylvester equations*, Electron. Trans. Numer. Anal., 43 (2014/15), pp. 142–162.

[13] P. Benner and J. Saak, *Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: A state of the art survey*, GAMM-Mitt., 36 (2013), pp. 32–52, http://doi.org/10.1002/gamm.201310003.

[14] D. A. Bini, S. Massei, and L. Robol, *On the decay of the off-diagonal singular values in cyclic reduction*, Linear Algebra Appl., 519 (2017), pp. 27–53, http://doi.org/10.1016/j.laa.2016.12.027.

[15] S. Birk, *Deflated Shifted Block Krylov Subspace Methods for Hermitian Positive Definite Matrices*, Ph.D. thesis, Wuppertal University, 2015.

[16] A. Bonnafé, *Estimates and asymptotic expansions for condenser p-capacities. The anisotropic case of segments*, Quaest. Math., 39 (2016), pp. 911–944, http://doi.org/10.2989/16073606.2016.1241955.

[17] S. Börm, *Efficient Numerical Methods for Non-Local Operators:* $\mathcal{H}^2$*-Matrix Compression, Algorithms and Analysis*, EMS Tracts Math. 14, European Mathematical Society, Zürich, 2010, https://doi.org/10.4171/091.

[18] D. Braess and W. Hackbusch, *Approximation of* $1/x$ *by exponential sums in* $[1, \infty)$, IMA J. Numer. Anal., 25 (2005), pp. 685–697, https://doi.org/10.1093/imanum/dri015.

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 2009.

[20] M. Crouzeix and C. Palencia, *The numerical range is a* $(1 + \sqrt{2})$*-spectral set*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 649–655, https://doi.org/10.1137/17M1116672.

[21] M. Dahleh, M. A. Dahleh, and G. Verghese, *Lectures on Dynamic Systems and Control*, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 2004.

[22] T. Damm, *Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations*, Numer. Linear Algebra Appl., 15 (2008), pp. 853–871, https://doi.org/10.1002/nla.603.

[23] E. D. Denman and A. N. Beavers, Jr., *The matrix sign function and computations in systems*, Appl. Math. Comput., 2 (1976), pp. 63–94, https://doi.org/10.1016/0096-3003(76)90020-5.

[24] H. C. Elman, K. Meerbergen, A. Spence, and M. Wu, *Lyapunov inverse iteration for identifying Hopf bifurcations in models of incompressible flow*, SIAM J. Sci. Comput., 34 (2012), pp. A1584–A1606, https://doi.org/10.1137/110827600.

[25] T. Ganelius, *Rational Functions, Capacities, and Approximation*, in Aspects of Contemporary Complex Analysis (Proc. NATO Adv. Study Inst., Univ. Durham, Durham, 1979), Academic Press, London, New York, 1980, pp. 409–414.

[26] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij, *Hierarchical tensor-product approximation to the inverse and related operators for high-dimensional elliptic problems*, Computing, 74 (2005), pp. 131–157, https://doi.org/10.1007/s00607-004-0086-y.

[27] A. A. Gonchar, *The problems of E. I. Zolotarev which are connected with rational functions*, Mat. Sb. (N.S.), 78 (1969), pp. 640–654.

[28] L. Grasedyck, *Existence of a low rank or* $\mathcal{H}$*-matrix approximant to the solution of a Sylvester equation*, Numer. Linear Algebra Appl., 11 (2004), pp. 371–389, https://doi.org/10.1002/nla.366.

[29] L. Grasedyck and W. Hackbusch, *A multigrid method to solve large scale Sylvester equations*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 870–894, https://doi.org/10.1137/040618102.

[30] L. Grasedyck, W. Hackbusch, and B. N. Khoromskij, *Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices*, Computing, 70 (2003), pp. 121–165, https://doi.org/10.1007/s00607-002-1470-0.

[31] L. Grubišić and D. Kressner, *On the eigenvalue decay of solutions to operator Lyapunov equations*, Systems Control Lett., 73 (2014), pp. 42–47, https://doi.org/10.1016/j.sysconle.2014.09.006.

[32] M. H. Gutknecht, *Block Krylov space methods for linear systems with multiple right-hand sides: An introduction*, in Modern Mathematical Models, Methods, and Algorithms for Real World Systems, Anamaya Publishers, New Delhi, India, 2007, pp. 420–447.

[33] A. Haber and M. Verhaegen, *Sparse solution of the Lyapunov equation for large-scale interconnected systems*, Automatica J. IFAC, 73 (2016), pp. 256–268, https://doi.org/10.1016/j.automatica.2016.06.002.

[34] W. Hackbusch, *Hierarchical Matrices: Algorithms and Analysis*, Springer Ser. Comput. Math. 49, Springer, New York, 2015, http://doi.org/10.1007/978-3-662-47324-5.

[35] M. Heyouni, *Extended Arnoldi methods for large low-rank Sylvester matrix equations*, Appl. Numer. Math., 60 (2010), pp. 1171–1182, https://doi.org/10.1016/j.apnum.2010.07.005.

[36] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[37] O. Kameník, *Solving SDGE models: A new algorithm for the Sylvester equation*, Comput. Econom., 25 (2005), pp. 167–187, https://doi.org/10.1007/s10614-005-6280-y.

[38] D. L. Kleinman, *On an iterative technique for Riccati equation computations*, IEEE Trans. Automat. Control, AC-13 (1968), pp. 114–115, https://doi.org/10.1109/TAC.1968.1098829.

[39] L. Knizhnerman and V. Simoncini, *Convergence analysis of the extended Krylov subspace method for the Lyapunov equation*, Numer. Math., 118 (2011), pp. 567–586, https://doi.org/10.1007/s00211-011-0366-3.

[40] J. G. Korvink and B. R. Evgenii, *Oberwolfach benchmark collection*, in Dimension Reduction of Large-Scale Systems, Lect. Notes Comput. Sci. Eng. 45, P. Benner, V. Mehrmann, and D. C. Sorensen, eds., Springer, New York, 2005, pp. 311–316, https://sparse.tamu.edu/Oberwolfach.

[41] D. Kressner and C. Tobler, *Krylov subspace methods for linear systems with tensor product structure*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1688–1714, https://doi.org/10.1137/090756843.

[42] V. Kučera, *Algebraic Riccati equation: Hermitian and definite solutions*, in The Riccati Equation, Springer, New York, 1991, pp. 53–88.

[43] I. Kuzmanović and N. Truhar, *Sherman-Morrison-Woodbury formula for Sylvester and T-Sylvester equations with applications*, Int. J. Comput. Math., 90 (2013), pp. 306–324, https://doi.org/10.1080/00207160.2012.716154.

[44] P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, UK, 1995.

[45] B. Le Bailly and J. P. Thiran, *Optimal rational functions for the generalized Zolotarev problem in the complex plane*, SIAM J. Numer. Anal., 38 (2000), pp. 1409–1424, https://doi.org/10.1137/S0036142999360688.

[46] J.-R. Li and J. White, *Low-rank solution of Lyapunov equations*, SIAM Rev., 46 (2004), pp. 693–713, https://doi.org/10.1137/S0895479801384937.

[47] S. Massei, D. Palitta, and L. Robol, *Solving rank-structured Sylvester and Lyapunov equations*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 1564–1590, https://doi.org/10.1137/17M1157155.

[48] D. Palitta and V. Simoncini, *Numerical methods for large-scale Lyapunov equations with symmetric banded data*, SIAM J. Sci. Comput., 40 (2018), pp. A3581–A3608, https://doi.org/10.1137/17M1156575.

[49] S. Pauli, *A Numerical Solver for Lyapunov Equations Based on the Matrix Sign Function Iteration in HSS Arithmetic*, Semester thesis, ETH Zurich, 2010, available from http://anchp.epfl.ch/students.

[50] T. Penzl, *Eigenvalue decay bounds for solutions of Lyapunov equations: The symmetric case*, Systems Control Lett., 40 (2000), pp. 139–144, https://doi.org/10.1016/S0167-6911(00)00010-4.

[51] J. K. Rice and M. Verhaegen, *Distributed control: A sequentially semi-separable approach for spatially heterogeneous linear systems*, IEEE Trans. Automat. Control, 54 (2009), pp. 1270–1283, https://doi.org/10.1109/TAC.2009.2019802.

[52] S. Richter, L. D. Davis, and E. G. Collins, *Efficient computation of the solutions to modified Lyapunov equations*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 420–431, https://doi.org/10.1137/0614030.

[53] E. Ringh, G. Mele, J. Karlsson, and E. Jarlebring, *Sylvester-based preconditioning for the waveguide eigenvalue problem*, Linear Algebra Appl., 542 (2018), pp. 441–463, https://doi.org/10.1016/j.laa.2017.06.027.

[54] J. Sabino, *Solution of Large-Scale Lyapunov Equations via the Block Modified Smith Methods*, Ph.D. thesis, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 2006.

[55] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288, https://doi.org/10.1137/06066120X.

[56] V. Simoncini, *Computational methods for linear matrix equations*, SIAM Rev., 58 (2016), pp. 377–441, https://doi.org/10.1137/130912839.

[57] C. F. Van Loan, *The ubiquitous Kronecker product*, J. Comput. Appl. Math., 123 (2000), pp. 85–100, https://doi.org/10.1016/S0377-0427(00)00393-9.

[58] Y. Xi, J. Xia, S. Cauley, and V. Balakrishnan, *Superfast and stable structured solvers for Toeplitz least squares via randomized sampling*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 44–72, https://doi.org/10.1137/120895755.

[59] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976, https://doi.org/10.1002/nla.691.

[60] J. Xia, Y. Xi, and M. Gu, *A superfast structured solver for Toeplitz linear systems via randomized sampling*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 837–858, https://doi.org/10.1137/110831982.